

An Environment for the Integrated Modelling of Systems with Complex Continuous and Discrete Dynamics

Manuel A. Pereira Remelhe¹, Sebastian Engell¹, Martin Otter², André Deparade¹, and Pieter J. Mosterman²

¹ Process Control Laboratory, Department of Chemical Engineering,
University of Dortmund, 44221 Dortmund
e-mail: {M.Remelhe, S.Engell, A.Deparade}@ct.uni-dortmund.de

² Institute of Robotics and System Dynamics,
DLR Research Center Oberpfaffenhofen, P.O.Box 1116, D-82230 Wessling
e-mail: {Martin.Otter, Pieter.J.Mosterman}@dlr.de

Abstract. The modelling and simulation of sophisticated technical systems is a demanding task. On the one hand, the physical part consists of a large number of subsystems which exhibit predominantly continuous dynamics, sometimes with (infrequent) discontinuities. On the other hand, the distributed computerised control systems constitute complex discrete-time and discrete-event systems that require completely different modelling and simulation methods. For an evaluation of the behaviour and the performance of the overall system, both types of models have to be combined and simulated efficiently. This contribution presents the requirements for a modelling environment for such systems and discusses an approach that consists of object-oriented modelling and efficient simulation of the physical part using the physical systems modelling language MODELICA, a software environment for the definition of discrete-event models using various formalisms, and the integration of both parts of the system via model translation. The coordination of both parts is performed by the MODELICA simulator. The modelling environment called DES/M (discrete-event systems for Modelica) supports the interoperation of different domain specific discrete-event formalisms. To illustrate the usage of the environment, a laboratory batch plant model is presented. A more elaborate example is described in another contribution in this volume (Mosterman et al., 2002).

1 Introduction

Sophisticated technological systems such as chemical plants, cars, and aircraft consist of a large number of physical components, numerous low-level set-point controllers and interlocks, and interacting complex supervisory controllers which may be organised in a hierarchical manner. On the supervisory control level, trajectory optimisation, fault detection, redundancy management, and sequence control e.g. for start-up and shutdown are performed and the interaction with the user is managed. The dominant part of the functions on this level consists of logic operations that are triggered by thresholds or events in the environment, including user commands. The physical part of the system and the supervisory control system put high demands on the power and the user-friendliness of the modelling techniques. In order to study the overall behaviour of such systems, a simulation model has to incorporate both

parts and an integration is required that enables efficient and at the same time accurate simulation.

1.1 The Physical Part

The physical part of the system consists of a large number of interconnected components. The behaviour of these components is determined by the laws of physics and chemistry. The overall system may consist of subsystems from various domains: electrical circuits, pneumatic and hydraulic actuators, mechanical transmission, fuel cells, combustion chambers, tanks, gas transport systems, chemical reactors, etc. These submodels are usually developed by teams of domain experts who take a lot of technological details and domain knowledge into account. Each modelling domain has specific graphical representations and modelling traditions, but in most cases the final models are algebraic and differential equations involving continuous variables that depend on (continuous) time. The models of the physical components may contain discontinuities that strictly speaking are caused by model simplifications which are made in order to avoid models with largely different time scales. Examples are friction and contact in mechanical systems, thermodynamic phase changes, ideal switches, e.g. diodes, in electronic systems. Other discontinuities occur when physical limits are reached (overflow of a tank, rupture of a vessel) or inputs to the physical system change abruptly. At these points in time, the movement of the system trajectory in the state space may abruptly change its direction and its velocity, or very fast transients occur that can be regarded as jumps in the state space. At points of discontinuity, the number of independent state variables may change, e.g. if two rigid bodies make contact. In consequence, the physical part of the system itself may exhibit hybrid behaviour, i.e. mixed discrete/continuous dynamics.

The complexity of modelling and simulation of the physical part of the system is exacerbated if several components with hybrid behaviour interact with each other, e.g. electronic circuits with several diodes. This calls for powerful modelling and simulation techniques.

1.2 The Supervisory Controllers

Supervisory control is used for many different purposes. For instance, sequential control is needed for the execution of recipes in chemical batch plants, redundancy management is crucial for the safety of aircraft, and resource booking systems are needed for coordinating several interacting sequential controllers, e.g., to avoid collisions of robots or to prevent the mixing of batches running in parallel in chemical plants. Start up, shut down, and emergency procedures are further examples for the necessity of supervisory controllers. In decentralised or redundant automation architectures, autonomous supervisory controllers interact in order to achieve the performance goals. Other functions of supervisory control are trajectory optimisation and user interaction.

In general, a supervisory controller is a reactive discrete-event system. The states and the outputs of such a system change discontinuously according to discrete state

transition sequences that are performed when a reaction to external stimuli from the user or from the physical system is required. For example, in the case of a tank that is being filled, a controller may have to close the inlet valve when the desired level is reached. The events that trigger the instantaneous reactions are determined by logical expressions containing analog and binary input variables that carry information on the current state of the physical system as well as internal variables that belong to the state of the controller and of other controllers in a distributed control system. Hence, the reactions depend on the current discrete state whereas the event times depend on the evolution of the state variables of the physical system and on clock variables in the discrete system. If the duration of a specific process, e.g. the duration of the filling of the tank, is known a priori and corresponding measurements, e.g., a sensor for the tank level, are not available, timers have to be used for triggering the transitions. Thus, time events occur that anticipate corresponding state events.

Even though supervisory controllers are mostly implemented as sampled data systems, their behaviour can adequately be described as reactive, i.e., driven by external state events. This is because the sampling intervals in the logical part are normally very short in comparison to the continuous dynamics so that at most sampling instants the controller does nothing but evaluating the triggering signals and waiting. Consequently, the sampling rate has a subordinate significance for the overall behaviour.

The difficulties for modelling and simulation arise from the fact that a reaction of a supervisory controller that appears as a monolithic state transition to the outer system may be the result of very complex inner iterations including hierarchical execution schemes as well as concurrency and synchronous and asynchronous communication.

1.3 Modelling and Simulation

The overall behaviour of a technical system is generated by the interaction of the physical components, discrete-event controllers and regulators. A precise comprehensive simulation model has to incorporate all these components and their relations if the purpose of the model is to evaluate the overall behaviour. Simulation goals may be, e.g., testing of the reaction to failures, the estimation of throughput or power consumption, a feasibility check for a specific production plan, or operator training.

Independent of the way of modelling, the physical part generally is solved by standard numerical integration methods such as Runge-Kutta methods or *backward differential formulae* (BDF) (Brenan and Campbell, 1996). This implies that the modelling process results in the generation of a consistent and uniquely solvable set of equations, either of explicit ordinary differential equations (ODE form) or of general *differential and algebraic equations* (DAE form). If hybrid phenomena have to be considered, special facilities have to be provided, because the inequalities that define the physical limits or the thresholds of a supervisory controller generate discontinuities, but the numerical integration methods usually require equations with a certain degree of continuity.

A usual approach is to ignore these inequalities during the numerical integration process and to use any efficient integration scheme, usually with a variable step size. This guarantees continuity of the equations. In order to handle the discontinuities, the values of the variables that enter into the trigger inequalities are monitored, and when a threshold is crossed, the integration is stopped and the time instant of the state event is localised up to a certain precision by backtracking. In case the event is dependent on time only, a time event, the integration simply stops directly at the predetermined time. When the integration is stopped, the discrete changes are performed and, afterwards, the integration is restarted.

The embedding of set-point controllers into the physical model is relatively straightforward since regulators are usually described by the same type of equations as the physical systems, and an overall ODE or DAE system results. Sampling effects often can be neglected because the sampling intervals are of the same order of magnitude as the integration step size. If this is not the case, time events have to be used in order to stop the integrator at every sample time. This is not convenient for multistep methods because these schemes must be restarted after every discontinuity which significantly decelerates the numerical integration (Brenan and Campbell, 1996).

In contrast to the domain of predominantly continuous dynamics where there is a standard system representation and general purpose numerical algorithms can be used, discrete-event simulation algorithms are specific to the modelling formalism used, and rather different from continuous integration methods. Popular formalisms are automata, statecharts, Petri Nets, dataflow diagrams, synchronous languages, or programming languages such as sequential function charts and function block diagrams as specified in the IEC 61131-3 standard for programmable logic controllers (IEC 1131, 1993). Each formalism has a specific syntax and semantics that closely matches users' training and expertise and that are well suited to the particular application. The transformation of formal models from one formalism into another is complicated and often leads to inefficient models, even for formalisms with equivalent expressive power (Huuck et al., 1997). Thus for a general purpose simulation environment, it is preferable, if not indispensable, to offer various modelling formalisms and even to allow the user to define new or specific formalisms with little effort. The use of domain specific formalisms results in models that are elegant, intelligible to the user, and closely correspond to the documentation formalism and/or the implementation language. This keeps the modelling effort low and makes it less error prone than a transformation into one general, tool-specific formalism.

2 Requirements for the Modelling Environment

Due to the complexity of both the physical part and the supervisory control system in large technical systems, it is evident that a powerful modelling environment and efficient simulation methods are indispensable to support the design process.

2.1 Intuitive and Effortless Modelling of Physical Systems

The physical part should be modelled as intuitively as possible. From the modeller's point of view the optimum would be to assemble the whole model using predefined building blocks that correspond to the technical components. The graphical connection of these elements would result in composition diagrams that look like familiar engineering notations, e.g. electrical circuit diagrams, flow charts, and other conventional notations.

In most cases, physical systems do not have explicit inputs and outputs; whether an external variable is input or output depends on the environment. For instance, the pressure drop in a pipe may be caused by a prescribed flow or be the cause of a certain flow rate. Thus the building blocks of larger models should have non-causal, undirected interfaces.

Due to the potential variety of components in technical systems, only a limited number of standard elements can be predefined and stored in component libraries. The remaining elements have to be defined by the modeller. For basic elements a convenient approach is to enter the underlying physical equations, possibly taken from the relevant literature, without transformation to a specific mathematical format, e.g. a system of explicit ODEs. Of course, the number of equations must match the number of unknowns. This approach is called declarative modelling, because the modeller states that these equations have to be satisfied, without determining how to perform the calculations. The model acts as a set of constraints on the coupling variables, but it is not explicitly stated how to compute outputs from inputs and initial states.

Larger elements should be defined as a composition of smaller building blocks. This leads to a hierarchical structuring of the model, which is crucial for the effective handling of large models. Since one has to deal with many different building blocks, it should be possible to establish user-defined libraries in addition to the standard libraries. Additionally, the concept of inheritance supports the modelling effort and reduces the likelihood of errors. Component model classes then can be derived from basic model classes by adding more detail. If the basic model class is modified, this modification also effects the derived classes and the models that will be instantiated from the derived classes.

2.2 Adequate Modelling of Discrete-Event Systems

The requirements for the modelling of discrete-event systems are different from those for physical systems in many respects. Discrete-event models are more diverse with respect to syntax and semantics than quantitative simulation models of physical systems. Physical systems can be treated in a uniform way using DAEs as an underlying semantic basis. The syntax of the graphical representation is also simple: the blocks have uniform ports and the building blocks are coupled by simply connecting these ports. In case of the modelling language MODELICA, the coupling semantics is that all (generalised) flow variables must add up to zero or that the (generalised) potential variables, such as voltage, pressure etc., assume the same value.

In contrast, each discrete-event formalism has its specific graphical syntax that does not simply refine a common basic syntax so that specific graphical editors have to be provided for each formalism that is supported.

Furthermore, no established semantic standard form, comparable to the DAE-system, exists for discrete-event formalisms, and transformations to a basic formalism are often inconvenient, sometimes due to relatively small semantic differences. Consequently, for the simulation of DES models, specific algorithms must be used.

Regarding the complexity of real supervisory control systems which may consist of a large number of modules that are specified by different designers from different domains, it is necessary to support heterogeneous discrete-event models including hierarchical execution schemes as well as concurrency with synchronous and asynchronous communication systems, i.e., it should be possible to model different parts of a controller with different formalisms and to connect these parts in a consistent manner. If different simulators are used for different formalisms, these discrete-event simulators have to interact with each other and have to be synchronised with the numerical integration of the continuous part of the system.

2.3 Integration of Continuous and Discrete-Event Models

For a seamless integration of discrete-event formalisms and physical models, on the *syntax* level, the coupling should reflect the actual hierarchical relations. Since components of the supervisory control system often are related to particular sub-systems of the continuous part, the corresponding discrete-event model should be represented by a block that can be inserted into a composition diagram of the physical model. The inputs and the outputs of the discrete blocks can be connected with ports of other building blocks, continuous or discrete.

On the *semantic* level, the coupling of a discrete-event model with the physical model is more involved. Some numerical integrators evaluate the model equations several times in order to do one step (Brenan and Campbell, 1996). This can cause unpredictable behaviours if the discrete-event system is called at intermediate points without considering the fact that the simulation of the continuous system has not yet converged. The semantics of the discrete-event formalisms must not become corrupted by the integration into the physical system. Conversely, transitions in the discrete-event part occur while time in the physical system does not progress. If iterations in the discrete-event part are performed, the intermediate states must not be transmitted to the continuous system but the simulation must be stopped until the discrete part has reached a stable state. This stable state may imply switchings not only of variables but also of the structure of the continuous part which may trigger new events in the discrete system. Even worse, the overall state of the continuous system, composed of the discrete inputs and the past state of the physical part may not be consistent such that a new initialisation has to be computed. So a nested loop of computations must be performed with frozen physical time until the overall system has reached a stable and consistent state from which the simulation can be continued.

The localisation of state events inevitably leads to increased simulation times because iteration or other additional computations are required. If the discrete part contains timers which trigger transitions, it is advantageous to propagate this information to the continuous simulator such that the simulation stops precisely at the event time and an iteration is avoided. Finally, discrete-event formalisms require an adequate visualisation of the simulation results using the graphical formalism itself typically in the form of animation. The usual plots of variables over time are not sufficient.

2.4 The State-of-the-Art

Some general-purpose commercial software tools exist for modelling and simulation of hybrid systems. Among these, the MATLAB package with SIMULINK and STATEFLOW is the most widely used tool (Matlab, 2002). In consideration of the requirements postulated above one has to realise that the state of the art is not satisfactory.

Block diagrams have a fixed causality and are not really intuitive to model large systems. The use of block diagrams results in an abstract mathematical representation of the modelled system as shown in Fig. 1. This block diagram corresponds to an electrical circuit, but it is not evident how it is related to the structure and to the parameters of the circuit. Furthermore, if e.g. a voltage source is replaced by a current generator many modifications are required in the overall model, since the cause and effect relations have to be inverted in several locations. This poses serious problems for the reuse of aggregated building blocks.

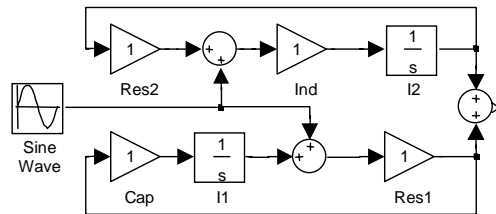


Fig. 1. A block diagram

The STATEFLOW formalism is a variant of statecharts (Harel, 1987). Figure 2 illustrates this with a STATEFLOW model of a relay mechanism. Statecharts are an intuitive and powerful formalism to model reactive behaviour and exist in many slightly different flavours. But besides statecharts, many other formalisms, such as High Level Petri Nets or GRAFCET, and programming languages, such as Sequential Function Charts, exist that have their specific strengths and can not be mapped easily onto statecharts.

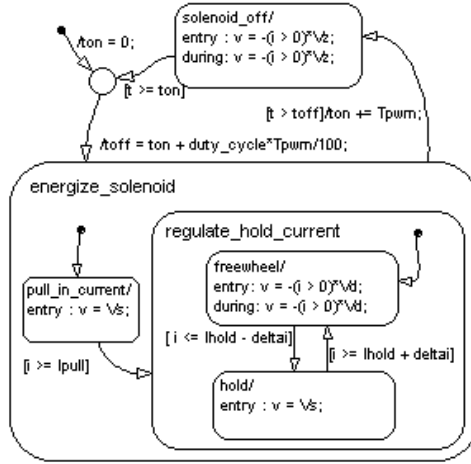


Fig. 2. A stateflow diagram (screenshot)

3 The DES/M Approach

The proposed solution for the modelling and simulation of large, complex systems with continuous and discrete-event dynamics consists of two major elements. The object-oriented equation-based modelling language MODELICA is used for the modelling of the physical part and of the regulators. A newly developed software tool for the modelling of discrete-event systems called DES/M (discrete-event systems for MODELICA) supports various formalisms (at present statecharts and SFCs) and modular, hierarchical and heterogeneous models. The discrete-event part of the model is automatically transformed into a MODELICA algorithm. Any simulator that can process MODELICA code can then be used to solve the overall system. The interaction of the continuous and the discrete part of the system is coordinated by the event handling mechanism of the MODELICA solver.

3.1 Object-Oriented Modelling Using MODELICA

In object-oriented modelling, the model elements correspond to physical components of the real system and the composition of the elements to the overall model is in accordance with the physical structure of the system. The elements have undirected interfaces and their behaviour is normally described declaratively. An element can be a composition of other elements and it can contain equations for the behavioural description. These equations need not to be solved explicitly for a particular variable. Another common feature of object-oriented modelling languages is that the equations are processed symbolically. The overall mathematical model is constituted by all the equations that describe the model elements and their connections. This usually leads to a large but sparse system of algebraic and differential equations (DAE). By means of automatic symbolic manipulations this large set of

equations is transformed into a sorted DAE where as many derivatives and algebraic variables as possible are computed explicitly and redundant variables are removed. This enables efficient simulation using standard integration methods.

The most prominent object-oriented modelling languages are MODELICA (Modelica Design Group, 2000), VHDL-AMS (Heinkel, 2000) and gPROMS (gPROMS, 2002). MODELICA is best suited for our purposes because it is not tailored to a specific application domain, and it is standardised by a non-profit organisation, the MODELICA Association, and freely available. Very important are the class concepts of MODELICA that include class definition, object instantiation, partial classes, inheritance, and more, which facilitate the creation of model libraries. These features are well known from object-oriented programming languages, but they are not always supported by object-oriented modelling languages. The meaning of the term ‘object-orientation’ depends on the context, here the essential property is the construction of large models from building blocks which can be used freely because they are formulated in a general, context-independent fashion. For MODELICA, many free libraries exist for different domains such as electrical systems, rotational and translational mechanics, multibody systems, and others.

For the definition and simulation of MODELICA models we use the commercial software DYMOLA (Dymola, 2002). This tool provides a graphical editor for composition diagrams so that systems can be modelled visually. The graphical representation of the library components mimics conventional engineering notations. The main reason to use DYMOLA, however, is the powerful symbolic engine that transforms the set of equations into a form that can be solved efficiently. This permits the simulation of very complex physical systems including hybrid phenomena (Otter et al., 1999).

In Fig. 3 it is shown how simple it is to build a model of a hydraulic actuator using given library components. The resulting model resembles the engineering notation and can be aggregated to a new composed building block that can be incorporated into a library as well.

To illustrate how hybrid phenomena can be modelled in an equation-based declarative style, consider an ideal electrical diode (Fig. 4). Due to the idealisation a sharp discontinuity is introduced at $u = 0$. In order to achieve an equation-based description, the diode characteristic is parameterised by a parameter s so that u equals s if s is less than zero, and i equals s if it is nonnegative. This results in the following set of equations:

$$off = s < 0 \quad (1)$$

$$u = \text{if } off \text{ then } s \text{ else } 0 \quad (2)$$

$$i = \text{if } off \text{ then } 0 \text{ else } s. \quad (3)$$

Due to this declarative formulation, the interaction of several diodes in an electrical circuit needs not be modelled explicitly. The network behaviour is defined implicitly by the composition of the component equations and of the connection equations (Otter et al., 2000).

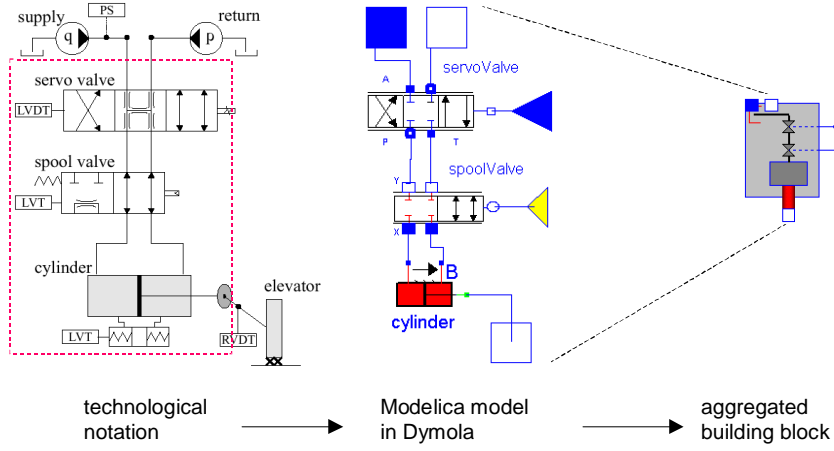


Fig. 3. Modeling a hydraulic actuator using standard components

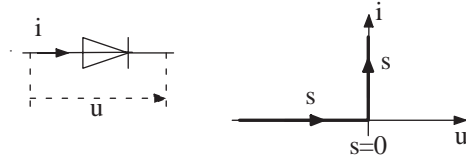


Fig. 4. Ideal diode model described as parameterized curve

Basic discrete event formalisms can also be expressed in an equation-based fashion, e.g. simple Petri Nets and automata (Mosterman et al., 1998). For instance, in a Petri Net model, the places and the transitions are represented by components that are defined in the corresponding MODELICA library. The graph structure is constituted by the connections of the ports of the components. Since each object and each connection just add equations to the overall set of equations, the behaviour of a Petri Net model is defined as the mathematical solution of the subset of equations given by the Petri Net model.

Unfortunately, this object-oriented modelling technique is not suitable for the modelling of complex discrete-event systems. The first reason is that the syntax of composition diagrams based on blocks with ports is not powerful enough for the graphical representation of complex formalisms such as statecharts. The second reason is that certain semantic elements such as local iterations can not be represented adequately by a set of equations. For instance, in certain statechart variants (Harel et al., 1987) a *step* of a statechart, i.e., its reaction to external stimuli, is defined as a sequence of *micro-steps*. Each micro-step consists of a set of concurrently taken transitions. At a micro-step, the firing transitions may generate events that trigger the transitions of the subsequent micro-step. In this manner a kind of event iteration is performed that ends when no further transitions are triggered (improper

statecharts may result in infinite iterations). Micro-steps are considered just as an internal mechanism to compute the reaction of a statechart so that the micro-steps should be hidden from the environment of the statechart. Therefore, an adequate realisation would use this operational semantics to generate the behaviour of a statechart and omit an interleaved execution with the physical system. Unfortunately this is not possible with an equation-based realisation, since the equations of a statechart would have to be solved simultaneously with the equations of the physical system. Thus, each micro-step would be connected to the evaluation of the overall set of equations so that side-effects possibly can take place in the physical system.

3.2 A Compatible Modelling Environment for Discrete-Event Systems

For the reasons stated above, the DES/M modelling environment has been developed that provides dedicated editors for several discrete-event formalisms and allows to insert the discrete-event models consistently into the overall model. By this approach the restrictions on semantics, syntax and graphical appearance are circumvented, and the object-oriented modelling principles for continuous systems are not enforced in a domain where they are not appropriate. By suitable transformations, the models of the discrete-event part can be inserted into the overall MODELICA model and can be solved using standard techniques for the manipulation and the numerical solution of continuous systems.

For the definition of the discrete-event part of the models, there are two different possible options. The first is to compose the model from discrete-event building blocks, the behaviour of which is specified declaratively based on equations, similar to the procedure that is followed for the continuous part. However, these blocks would have to be quite complex because a large number of interacting variables may be required. Therefore the blocks should not simply be merged but a code optimisation step should be performed. Thus there would be two transformations before an executable model is obtained; first the transformation of the individual blocks into MODELICA code, then the construction of the overall model. The second approach is to construct the discrete-event part of the model completely on the graphical level using the chosen formalisms and the respective graphical editors, and then to perform an automatic translation into a single MODELICA-algorithm and to wrap it into a MODELICA class. We prefer the second approach. For reasons discussed above, all discrete-event subsystems that interact directly via events or messages must be represented as a monolithic block in an imperative fashion. The transformation of the complete system into an algorithm leads to a clear structure – first an overall discrete-event model is composed from subblocks that can be structured hierarchically and may even be defined using different formalisms, e.g. statecharts and SFCs, and then the transformation into an algorithm is performed following clearly specified semantics. Actually, in the end a problem specific discrete-event simulator is inserted into the MODELICA model of the physical system. This MODELICA component can be easily connected to physical components because it interacts via standard ports.

The main advantage of using a MODELICA-algorithm is that the handling of the state events is done by MODELICA automatically. The MODELICA compiler discovers all potential sources of discontinuities in the algorithm and makes sure that discontinuities are handled appropriately, i.e., when a threshold is reached and a discrete state transition or any other discontinuity occurs, the integrator will be stopped in order to perform the discrete changes. If the discrete-event model would be simulated by an external program, the conditions that trigger the state transitions in the discrete-event model still would have to be inserted into the MODELICA model in order to stop the continuous simulation when the discrete-event part causes state events. If the discrete system is specified in a different environment, this task has to be performed manually by copying the transition conditions or guards and invariants, which is tedious and error-prone. In contrast, the DES/M environment generates automatically a complete MODELICA simulation algorithm for the discrete system parts from the graphical specification.

The modelling environment supports heterogeneous and hierarchical discrete-event models by means of a special block editor. Model-reuse is enabled using an archetype concept, i.e., each block that is used in a model is an instance of an archetype that defines the ports and the general properties of the block type and one or several alternative implementations. These implementations define the behaviour of the instantiated blocks and can be specified using again block diagrams or another formalism.

In order to reduce the effort for the implementation of several editors, the DES/M environment is based on the meta-modelling tool DoME (DoME, 1999). DoME was designed as a tool for the automatic generation of complex graphical editors based on a formal syntax description and parameters that control the graphical appearance. A partially graphical language called DoME Tool Specification Language is used for specifying the graphical entities, their properties and relations, structural constraints as well as their visual appearance. More advanced features such as more complex syntactical constraints and code generation can be implemented with DoME's Lisp-like extension Alter or using Smalltalk. Besides the block diagram editor, up to the present, two further editors have been realised: a statechart (SC) editor and an editor for *sequential function charts* (SFC) (Deparade et al., 2001).

3.3 Formalism Interoperation via Special Block Diagrams

As already mentioned, a special hierarchical block diagram formalism has been implemented for supporting the interoperation of different formalisms. The main idea is rather straightforward: Certain blocks of a block diagram may contain either another block diagram or a reactive model that is specified with a state transition formalism such as statecharts or sequential function charts. Consequently, it is possible to use different formalisms within one model.

The idea to use a block diagram formalism arose from the modelling of the aircraft elevator described in detail in (Mosterman et al., 2002). The main feature of this control system is that 8 concurrent state machines, each modelled by a statechart, interact tightly in order to achieve a safe configuration of the redundant el-

evator actuators when failures occur. The statecharts have the same structure and their transition conditions are large logical expressions that reference the states of the other statecharts and the failure signals. The goal of the block diagram formalism was to separate the large and complex logical expressions from the statecharts, so that the statecharts become identical (and clearer) and can be instantiated from the same class. Therefore, the block diagram formalism distinguishes static blocks that are depicted with a dashed border, from dynamic blocks that have a solid border (Fig. 5).

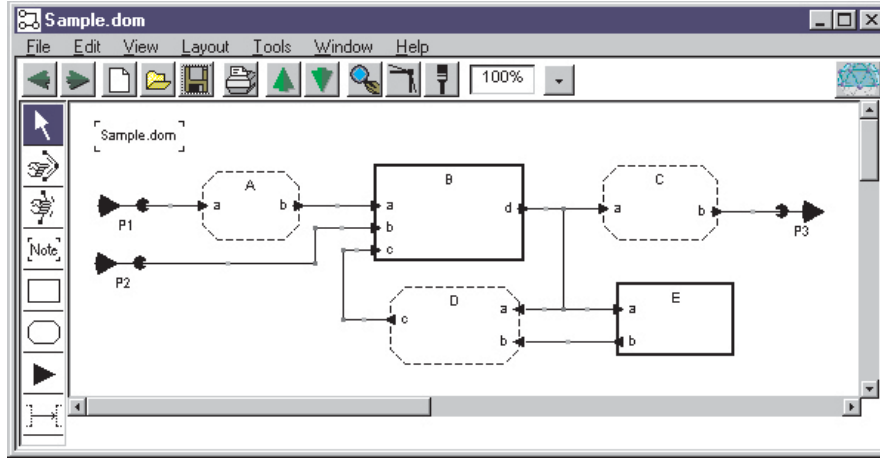


Fig. 5. A sample block graph

A static block contains an algorithm or just a set of assignments and is used to compute the current output values y_i directly from the current input values u_i of the block. Hence the behaviour of a static block can be represented by a function:

$$y_i = f_{stat}(u_i) \quad (4)$$

Such a static block is applied to, e.g., the computation of the logical expressions of the redundancy controller.

The dynamic blocks have internal state variables x_i and a quasi-synchronous semantics is applied, i.e., the blocks are evaluated synchronously, but without simultaneous data exchange:

$$x_i = f_{dyn}(x_{i-1}, u_i), \quad (5)$$

$$y_i = g_{dyn}(x_{i-1}). \quad (6)$$

The state transition function f_{dyn} and the output function g_{dyn} impose an iterative computation scheme for the block graph such that the response of such blocks to new changes of the inputs becomes effective in the next iteration step. As long as

the outputs of these blocks are changing, all blocks have to be reevaluated synchronously. This quasi-synchronous semantics is analogous to the internal computation of statechart behaviour: if a statechart contains orthogonal parts (modelled with and-states), the consequences of concurrently and independently taken transitions of a micro-step, i.e., events and the new states, only become effective in the subsequent micro-step. Thus, in the DEFORM approach, local event iterations are not only performed inside of the statecharts, where a step can be computed by a sequence of micro-steps, but also on the block diagram level.

Further elements in Fig. 5 are the outer ports that represent the interface of the block diagram to the higher level ($P1, P2, P3$) and the ports of the blocks (a, b, c, d). Each port has an associated port type that defines the structure of the data transmitted through the respective port. This data-structure can be hierarchical and may contain different basic types such as Real, Integer and Boolean.

At a higher level, the block diagram in Fig. 5 is itself a dynamic block with ports $P1, P2$ and $P3$. The state of this enclosing block is the Cartesian product of the states of the dynamic blocks B and E . For the computation of the state transition function of the enclosing block an iteration at the level of the inner block graph (Fig. 5) is started during which the following constraints have to be satisfied at each iteration step:

$$\begin{aligned}
A.a_i &= P1 \\
B.b_i &= P2 \\
E.a_i &= D.a_i = C.a_i = B.d_i = g_B(B.x_{i-1}) \\
D.b_i &= E.b_i = g_E(E.x_{i-1}) \\
A.b_i &= f_A(A.a_i) \\
C.b_i &= f_C(C.a_i) \\
D.c_i &= f_D(D.a_i, D.b_i) \\
B.a_i &= A.b_i \\
B.c_i &= D.c_i \\
B.x_i &= f_B(B.x_{i-1}, B.a_i, B.b_i, B.c_i) \\
E.x_i &= f_E(E.x_{i-1}, E.a_i).
\end{aligned} \tag{7}$$

After this iteration has converged to a stable state, the outputs $P3$ of the enclosing block are updated and the computation of the transition function of the enclosing block is finished.

It should be noted, that for a specific block it does not make a difference whether its behaviour is specified as a block diagram or as a statechart, since both formalisms are transformed into a state transition function that hides the inner processes. Hence arbitrary other reactive formalisms and communication paradigms can be incorporated as well, as long as they can be transformed into a compatible state transition function.

3.4 The Modelling and Simulation Process

The approach described above leads to a tool architecture that consists of two main cooperating tools: DYMOLA is used for physical system modelling, whereas the

DES/M environment is used for modelling discrete-event systems (Fig. 6). By means of the editors for the various discrete-event formalisms, the complete supervisory control system is described. Then it is compiled into a MODELICA class that is stored in the file system so that it can be retrieved by DYMOLA and instantiated in the model of the physical system. The MODELICA classes created in DYMOLA are stored in the file system as well. For simulating the overall model, the corresponding class has to be compiled into an executable. The transformation of the set of equations into a preferably explicit representation is performed automatically. The simulator executable generates the trajectory for a given set of parameters that can be changed without the need to recompile the model. Every time when the supervisory controller has to react, the integrator stops because a state event is generated due to inequality expressions in the MODELICA-algorithm. The execution of the algorithm at these times realises a discrete state transition and the corresponding change of the outputs. The internal processes during such a state transition do not become visible to the model of the physical system, but they are saved in a log file. This permits the visualisation of the internal processes of the discrete-event model in the DoME tool for debugging purposes.

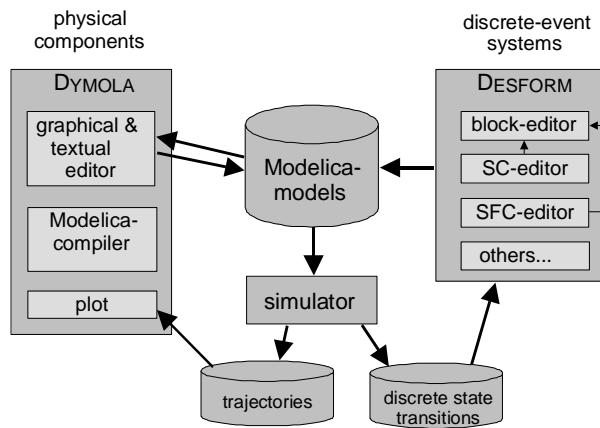


Fig. 6. The modeling and simulation process using two tools

4 Realising Discrete-Event Dynamics in MODELICA

A discrete-event model that was composed within the DES/M environment is translated into a MODELICA component that contains one algorithm for the computation of the reactions of the corresponding supervisory controller. This algorithm is a simulator for the specific discrete-event model and is possibly very complex. In the following, two simple examples are discussed in order to illustrate how the continuous integration and the discrete-event dynamics are combined using the MODELICA

language. The actual code generation is intricate, in essence it is the realisation of the operational semantics of the formalisms supported by DEFORM using the MODELICA language.

4.1 Models with State Events

The synchronisation of the discrete-event dynamics and the continuous integration is straightforward (Pereira Remelhe et al., 2001). To illustrate this, consider a simple supervisory controller that fills a tank up to a certain level h_high , after a specific low level h_low was reached. For safety reasons, an additional limit sensor is installed that indicates whether the tank is full. This controller has two input variables: the current level h and the binary signal $limit_h_full$, as well as a binary output variable v for the inlet valve. The corresponding discrete-event dynamics can be described by a model with two states $S1$ and $S2$, and two Transitions $T1$ and $T2$ (Fig. 7). An algorithm that exhibits the desired behaviour can be formulated as follows:

```
T1_fires := pre(S1) and (limit_h_full or (h>h_high));
T2_fires := pre(S2) and (h<h_low);
S1 := (pre(S1) and not T1_fires) or T2_fires;
S2 := (pre(S2) and not T2_fires) or T1_fires;
v := S1;
```

Such an algorithm corresponds to a state transition function that maps the previous controller state $x_{prev} = \{pre(S1), pre(S2)\}$ and the current input variables $u = \{h, limit_h_full\}$ into the new state $x_{new} = \{S1, S2\}$ and the current output variables $y = \{v\}$. During the continuous integration this algorithm is executed simultaneously to the evaluations of the complete set of equations, but all state variables and outputs remain constant, since the inequality expressions are fixed. If the state $S1$ is active, in the algorithm $pre(S1)$ is true and $pre(S2)$ is false. Therefore only the first logical expression can become true, and, consequently, only the inequality expression $(h>h_high)$ needs to be monitored during continuous integration. When this expression becomes true, the integrator is stopped and the whole set of equations including the algorithms is re-evaluated including the unfixed inequality expressions. Now the value of $T1_fires$ becomes true, $S1$ becomes false, $S2$ becomes true, and v becomes false, i.e., the state changes from “filling” to “waiting”. In a second discrete evaluation only the transition variable $T1_fires$ becomes false again, since $pre(S1)$ is now false. Because the discrete state variables did not change this time, the integration is started again. Now $(h<h_low)$ is monitored.

4.2 Models With Time Events

As an alternative, the limit sensor could be replaced by a time-out corresponding to the known maximum duration of the filling process. This idea is realised in the

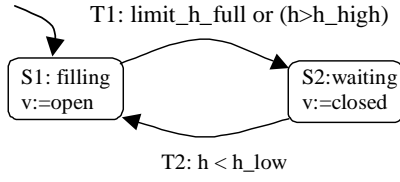


Fig. 7. Discrete-event model using only state events

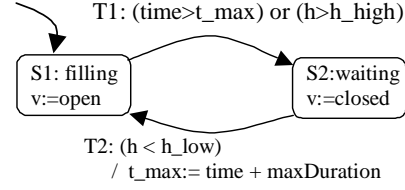


Fig. 8. Discrete-event model using state events and time events

diagram shown in Fig. 8. When the transition T2 is taken, an action is performed that assigns a new value to the variable t_max that stores the point in time, when the state S1 has to be left. Additionally, the transition T1 makes sure that the filling activity stops when this time elapses. A corresponding algorithm is as follows:

```

T1_fires := pre(S1) and ((time>pre(t_max)) or
(h>h_high));
T2_fires := pre(S2) and (h<h_low);
t_max := if T2_fires then time+maxDuration else
pre(t_max);
S1 := (pre(S1) and not T1_fires) or T2_fires;
S2 := (pre(S2) and not T2_fires) or T1_fires;
v := S1;
  
```

Hence, an additional state variable t_max is needed that, in contrast to the other state variables, is a real valued variable. If the controller is in state S1 the inequality expression $(h>h_high)$ has to be monitored in order to generate a state event, but as long as the choice of t_max is correct, the expression $(time>t_max)$ is used to generate a time event and the simulation stops exactly at the corresponding time without the need to localise a state event.

5 An Illustrative Application Example

To illustrate how the DES/M environment can be applied, a model of a laboratory batch plant is presented that incorporates hybrid physical dynamics and a supervisory controller. The plant is a slightly simplified variant of one of the benchmark examples in this volume and was already described in (Kowalewski and Preußig, 1996). The physical part of the plant has been modelled in an object-oriented and equation-based fashion using the MODELICA language. A library has been developed that provides the classes *Valve*, *Pump*, *Condenser*, *Sensor* and 4 different types of tanks. These were graphically composed in the DYMOLA tool resulting in a process flow chart (Fig. 9) that resembles the graphics of a standard piping and instrumentation diagram.

The supervisory controller model is also included in the plant model, but the sensor objects and the actuator objects are not connected visually to the controller

component inputs or outputs respectively, in order to keep the model clear. Instead, on the top level of the model, additional equations are used that relate the current values of the sensors to the input variables of the input port of the controller, e.g.:

```
controller.sensors.LIS_101 = LIS_101.value;
```

or that relate the input signals of the actuators to the outputs signals of the controller, e.g.:

```
V1.open = controller.actuators.V1;
```

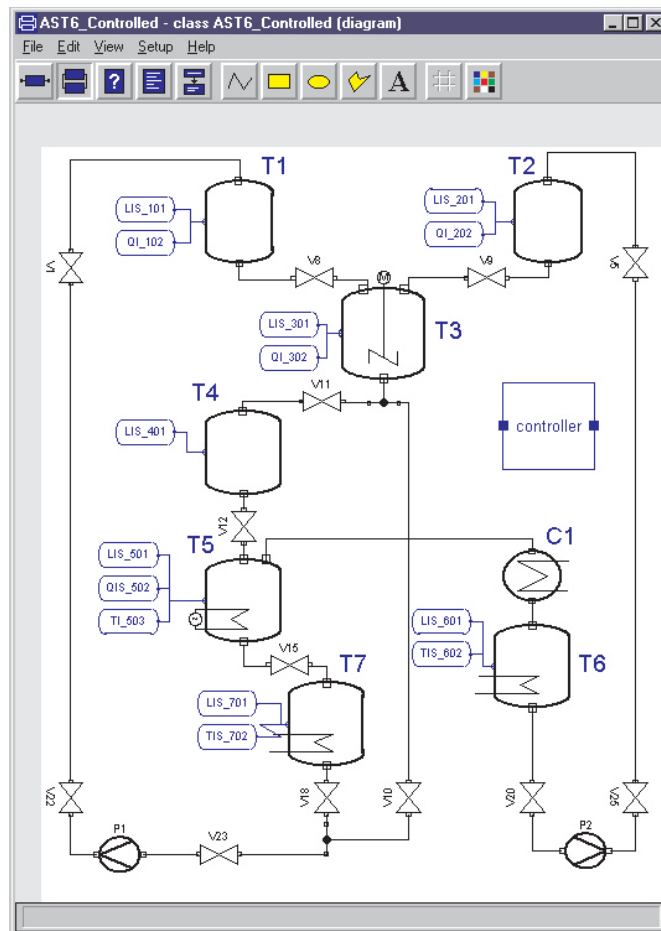


Fig. 9. The MODELICA model of the batch plant

The DES/M environment generated the MODELICA class of the supervisory controller from a graphical specification that includes sequential function charts (SFC)

and the block graph formalism. Figure 10 shows the overall structure of the controller model. The objective of this controller is to run 2 recipes in parallel on the plant. As a rudimentary means of coordination, the idle tanks are determined from the sensor and actuator values using simple logical expressions such as:

```
idleTank.T7_idle := (sensors.LIS_701<=0.001) and
not (actuators.V15) and not (actuators.V18);
```

This means that a tank is idle if the tank is empty and the connected actuators are passive. Since both recipes drive the same actuators, their outputs have to be superposed. This can be realised by simple logical expressions that activate an actuator, if one recipe (or both recipes) set this actuator active, e.g.:

```
actuators.V1 := act1.V1 or act.V2.
```

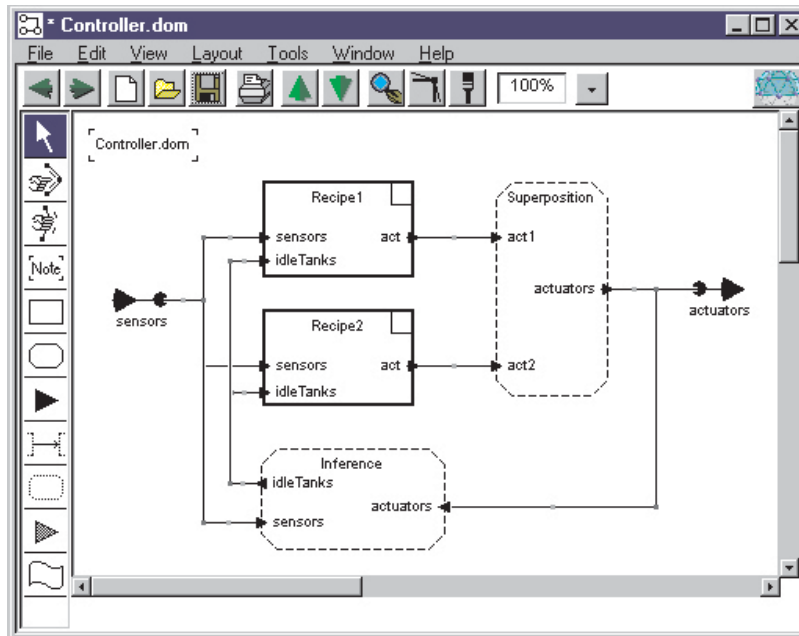


Fig. 10. The top level block graph of the controller

These logical expressions were entered as static blocks. The only dynamic discrete-event behaviour results from the recipes that are contained in the dynamic blocks. Both recipe blocks run the same recipe and only differ in their parameters such as the start time. Therefore, both recipes are described by the same SFC, see Fig. 11. When the start time of a recipe elapses, an infinite loop is started where a concentrated salt solution of tank T1 is drained into tank T3 and then mixed with pure water from tank T2 until a certain concentration $w_dilution$ is obtained.

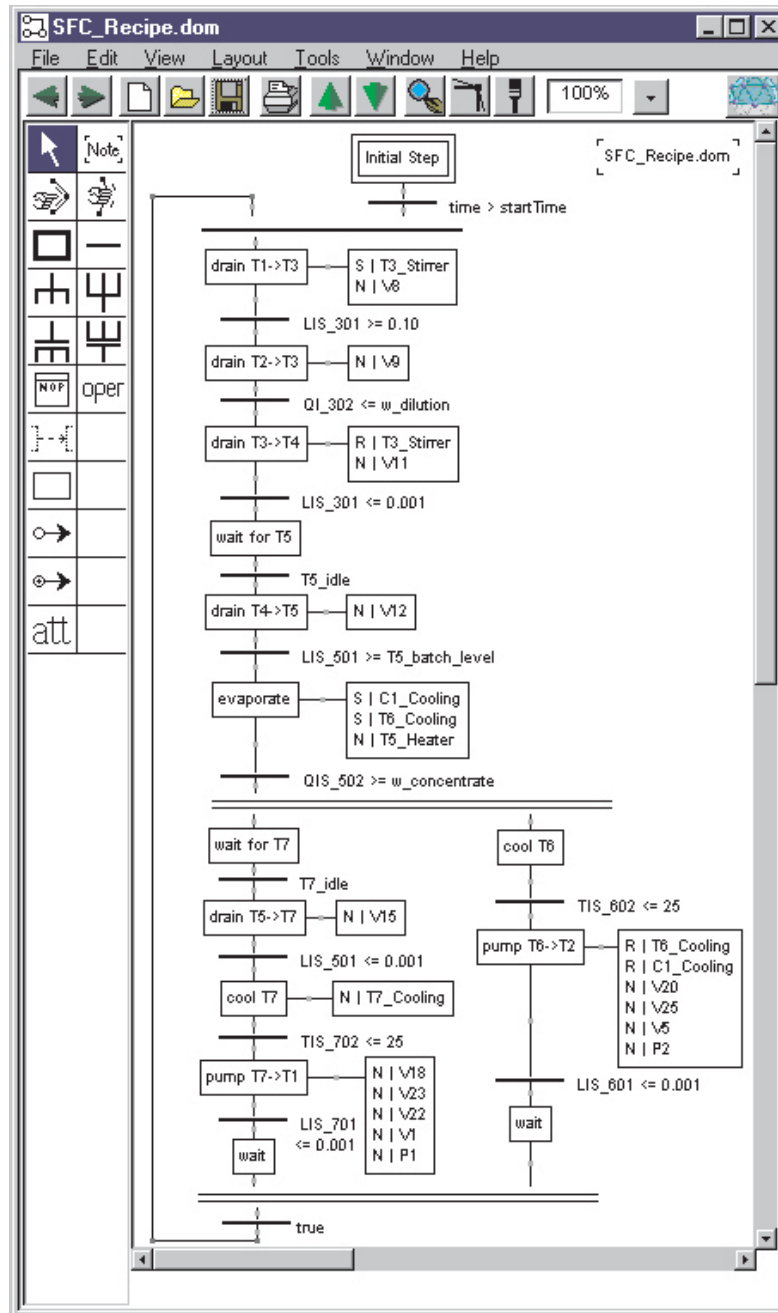


Fig. 11. The sequential function chart of the recipes

This dilution is buffered in tank T4 and then concentrated in the evaporator T5 up to a certain concentration $w_concentrate$. The condenser and the cooling in tank T6 are started at the same time at which the electrical immersion heater of T5 is started. When the evaporation is stopped, the recipe splits into 2 parallel threads in which the concentrate is drained into tank T7 and cooled and then fed back to tank T1, and the condensate in tank T6 is cooled and fed back to tank T2. Then the recipe returns to the first step and repeats this procedure.

In order to omit redundancies in the model, types were declared for ports and blocks. A port type defines which variables are transmitted through a port and their numerical types (Real, Integer, or Boolean). In this example, only three port types are needed: *Sensors*, *IdleTanks*, and *Actuators*. Each port in the model must have an assigned type and connected ports must have the same type. The connections are checked automatically before the translation to MODELICA.

Since all recipes in this controller framework must have the same interface, an archetype is defined for the recipe blocks. Figure 12 shows the interface of the archetype in the centre. The interface defines the ports and their port types for all possible recipe blocks. In the upper right region of the window, both instances of this archetype are enumerated corresponding to the blocks of Fig. 10. In the lower right region, the behaviour specifications that are possible implementations of this archetype are listed. In this case only one SFC is given. But more SFCs can be specified for the usage of different recipes. In principle, also statecharts and block graphs are possible as implementations of a recipe block.

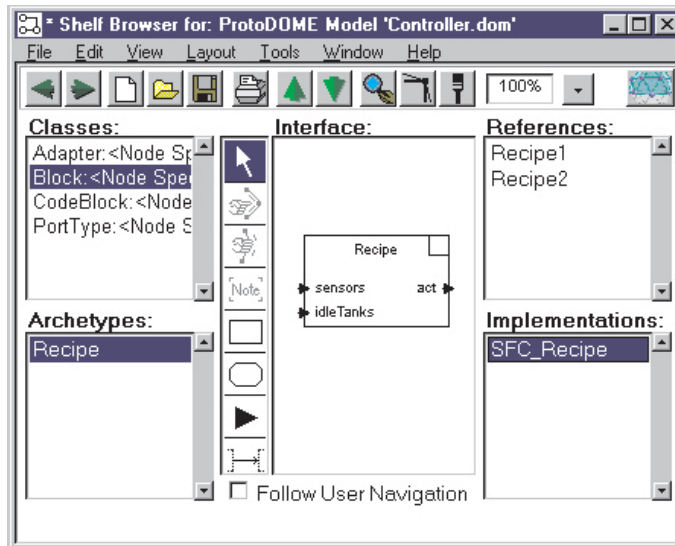


Fig. 12. The Shelf Browser in DOME with the archetype for the recipe blocks

After finishing the overall controller model, the MODELICA class can be generated automatically by selecting a corresponding item from a pull down menu. The MODELICA class contains a data structure that corresponds to the structure of the controller and an algorithm that defines the behaviour. This class can be instantiated in the plant model just like all other components. For simulation, the MODELICA model is compiled with DYMOLA resulting in a simulator executable. The model parameters can be changed conveniently for each simulation run without the need to recompile the model. This includes also the parameters of the controller such as the start times and the concentration parameters of each SFC. Thus, parameter studies can be performed efficiently. Figure 13 shows the plot of the mass hold-up of the evaporator as it results from a simulation run. From this plot one can conclude that the alternating recipes operate the evaporator at full capacity, i.e., there are no gaps where the evaporator is waiting for the next batch.

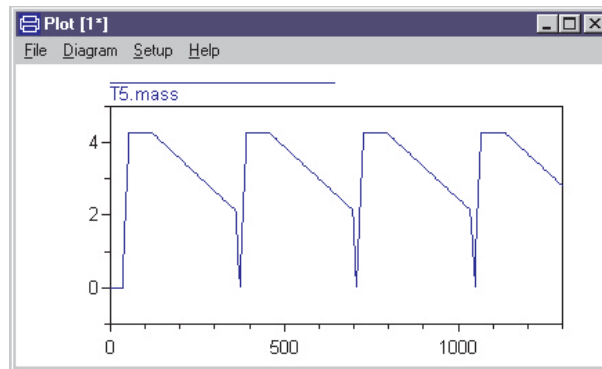


Fig. 13. A plot of the mass hold-up of the evaporator T5

6 Conclusions

The different nature of physical systems and supervisory control systems leads to two distinct sets of requirements for an integrated modelling and simulation methodology. In this work, the physical part of a system is captured by the object-oriented modelling language MODELICA is used, and for the discrete-event parts the DES/M environment has been developed.

MODELICA permits a very intuitive modelling of physical systems even if they include hybrid phenomena, and meets the demands for object-oriented modelling and effective and efficient simulation. The DES/M environment provides dedicated editors for domain specific discrete-event formalisms (and programming languages), and supports heterogeneous models including hierarchical structures and interoperation.

The connection of both parts of a model is done by translating the complete discrete-event model into a MODELICA algorithm that is wrapped in a MODELICA class. This is advantageous in several respects:

- both modelling environments are integrated seamlessly,
- the operational semantics of discrete-event formalisms can be formulated in an arbitrary way and this semantics is realised without any disturbance by the interaction of the continuous and the discrete part of the model or the solution algorithm of the continuous part (e.g. the step size of the integrator is independent of the execution of discrete transitions which occur at frozen simulation time),
- state events are accessible from the discrete-event part.

The latter point is crucial for an accurate synchronisation of continuous integration and discrete-event simulation.

At present, a first prototype of the DES/M environment is available. The current work aims at the improvement of the translation framework towards a more general approach and on the automatic support for graphical data visualisation. Then more formalisms will be implemented. Since MODELICA is based on algebraic and differential equations also other hybrid formalisms such as hybrid Petri Nets can be considered.

Acknowledgement

We thank Hilding Elmqvist and Johann Bals for the helpful discussions. The research reported here was performed in the context of the focused research program (Schwerpunktprogramm) “Continuous-Discrete Dynamic System” (KONDISK) and sponsored by the Deutsche Forschungsgemeinschaft under the grants OT174/1 and EN152/22. This support is most gratefully acknowledged.

Simulation for Analysis of Aircraft Elevator Feedback and Redundancy Control

Pieter J. Mosterman¹, Manuel A. Pereira Remelhe², Sebastian Engell², and Martin Otter¹

¹ Institute of Robotics and Mechatronics, DLR Oberpfaffenhofen,
P.O.Box 1116, D-82230 Wessling, Germany

² Process Control Laboratory, Department of Chemical Engineering,
University of Dortmund, 44221 Dortmund, Germany

Abstract. Safety critical systems such as aircraft require functional and hardware redundancy to achieve prescribed safety levels. Discrete event control is applied to ensure that a safe system configuration is available at all times. Since, at present, formal verification techniques are restricted to models with few continuous states, in this paper, simulation is used to verify that the overall system operates according to the requirements when an actuator failure occurs. The feasibility study to modelling and simulation of complex controlled systems presented here is characterised by (i) a complex object-oriented model of aircraft dynamics, including gravity, aerodynamics, etc., (ii) the specification of the discrete event redundancy control by a domain specific formalism that includes statecharts, (iii) the usage of energy based hybrid bond graphs to model the dynamics of the hydraulic actuators, (iv) model integration on the model level as well as on the data level, (v) support of DAEs with dynamically changing index and (vi) illustrative simulation results.

1 Introduction

Redundancy is one of the most important techniques to achieve the desired level of safety in systems such as aircraft, nuclear plants, chemical plants, and other safety critical applications. Its basic premise is to include redundant functionality into a system that can be activated when failures of the normal operating components occur and to validate and select normal behaviour (e.g., voting procedures).

1.1 Aircraft Attitude Control

To illustrate the concept, consider the primary (attitude) control surfaces of an aircraft as shown in Fig. 1. The ailerons are used to control roll, the elevators control pitch, and the rudder controls yaw motion. This paper concentrates on the pitch control, performed by the elevators. Each of the elevators is positioned by one of two actuators, the other one operates as a passive load. Discrete-event control embedded on two primary flight control units (PFCU) selects the controlling actuator and ensures that the redundant actuator is loading. Each PFCU controls one actuator per elevator, so that both elevators can be controlled, even if one PFCU fails completely. The PFCUs also generate the position control signals for the four actuators.

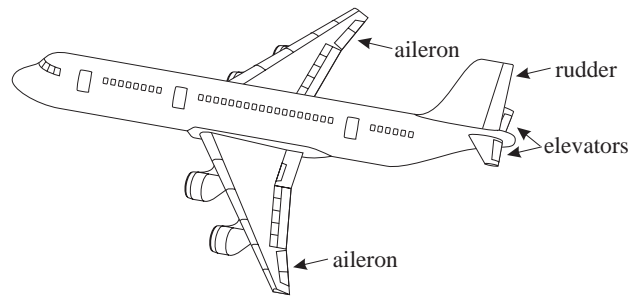


Fig. 1. Primary control surface of an airplane

Feedback control is used for normal operation whereas so-called direct link control is applied to single actuators in the case of certain failures. To ensure minimal transient disturbances caused by actuator switching, the loading actuator should be shadowing the control signals ready to switch to a mode where it actively controls the elevator. However, in some extreme cases the actuator may be disengaged, i.e., it is loading but not shadowing. Thus, each PFCU has to decide for two actuators whether an actuator is disengaged, shadowing, or controlling and whether a feedback or a direct link controller is used for shadowing or controlling. These decisions depend on the mode of the other actuator, the state of the other PFCU and the detected failures. The best possible consistent state configuration of both PFCUs for a given failure situation is achieved by a complex iterative interaction of both PFCUs.

The hydraulic actuator design and the controller parameters may influence the overall behaviour of the aircraft significantly. Therefore, all contributing parts and phenomena of the aircraft such as aerodynamics, gravity, engines, etc. have to be considered in order to assess the design of the elevator control system. Because of the immense complexity and the intricate redundancy management model-based validation is required.

Formal verification techniques are widely used for pure discrete-event systems and much research has been carried out recently on the verification of hybrid systems. However, at present, the complexity of systems amenable to hybrid systems verification techniques is restricted to a low order continuous dynamics (typically not more than three continuous state variables) (Benedetto and Sangiovanni-Vincentelli, 2001, Lynch and Krogh, 2000, Vaandrager and van Schuppen, 1999). Consequently, formal methods are applied to the discrete-event part only, e.g., a so-called Failure Mode Effect (FME) analysis is employed to verify certain safety and reliability properties of the redundancy management system. However, its interaction with the continuous parts as well as the design of the position controllers and the hydraulic actuators can not be evaluated with formal verification techniques. Therefore, the only practical model-based approach for this task is to perform extensive simulation studies.

1.2 Model Design

In this contribution, we concentrate on the modelling and simulation of the elevator control system and the aircraft. The model formulation is driven by the assumption that the simulation studies have the purpose to assess whether the design of the evaluator control system meets the requirements with respect to the overall behaviour of the aircraft (e.g., lateral and longitudinal aircraft velocity and flight path angle). In particular, different sets of parameters of the controllers and of the hydraulic actuators have to be tested in combination with certain failure scenarios.

As a consequence, the simulation model has to incorporate a realistic model of the aircraft dynamics, including all essential effects and components such as aerodynamics, gravity, engines, and hydraulic oil supply. In order to automatically generate the correct Boolean input signals of the feedback controllers and the actuators depending on the sequence of failure events it is convenient to include at least the input-output behaviour of the redundancy management components. Since the sampling times of the PFCUs are very fast in comparison to the bandwidth of the actuators, the hardware aspects of the PFCUs can be neglected, i.e., the redundancy management model reacts instantaneously on failures and the controllers are modelled as ideal continuous controllers. Another idealisation is introduced for the hydraulic actuators. There are many small physical effects such as oil elasticity, viscosity, and fluid inertia which do not influence the overall dynamics significantly, but considerably increase the modelling effort, so that these effects are not considered in the corresponding models.

These basic model design decisions cause several difficulties for the modelling and the simulation. With respect to modelling, the complexity of the systems and their heterogeneous nature mandates the use of dedicated formalisms. These formalisms differ greatly in their visual representation and require the interoperation of specific and powerful modelling environments.

Present day simulation technology, on the one hand, can handle large systems of differential and algebraic equations (DAE), possibly extended by some discontinuous equations (ABACUSS, 1995). On the other hand, discrete-event simulators apply an event driven approach to manage the huge number of state changes in discrete-event models (Group, 1999). The combination of discrete and continuous behaviour requires the integration of a numerical integrator with some sort of discrete-event simulation. Especially, the detection and location of discrete events during continuous integration has to be supported. Furthermore, at event times discontinuities in continuous state variables may occur. For the aircraft model, this phenomenon emerges because the abstractions in the hydraulic actuator models result in a DAE with dynamically changing index. This requires a special simulation engine that switches the active equations and automatically reinitialises the state variables according to physical conservation laws, when the index changes. This contribution presents techniques that cope with all these problems.

1.3 The Modelling and Simulation Approach

For the components of the physical system we use object-oriented modelling. In this context, the term object-oriented modelling means that every physical object is modelled independently without making assumptions about its environment and preserving the physical connection structure of the object. The connections of a model component have to correspond to physical interactions the computational causality of which is not fixed a priori, i.e., the variables involved in an interaction are not a priori defined as inputs or outputs. Furthermore, the behaviour of the component should be defined in a declarative way where a set of (possibly implicit) equations is regarded as a set of behavioural constraints rather than as a calculation formula. To illustrate this, let us consider a hydraulic line. The component model of the line would have two connections which each incorporate a pressure and a flow variable. These variables represent neither inputs nor outputs, since depending on the structure of the environment the pressure drop causes the flow or the flow causes the pressure drop. In some cases the causality can even change dynamically so that a quantity that would be regarded as an input in signal flow diagrams becomes an output and vice versa. This is why the equation-based behavioural description is inherent to object-oriented modelling. Using equations (which may be written in an implicit style) for the description of the behaviour does not impose a specific calculation scheme. From the modelling perspective the equations of all model components and all connections of the overall aircraft model simply form a global set of differential and algebraic equations (DAE) so that simulation is the task to find a solution to these equations, i.e., functions over time that satisfy the equations. To generate efficient simulation code, the model equations must be processed by a symbolic engine and compiled into executable code.

The existing DAE based modelling languages such as gPROMS (Barton, 1992), VHDL-AMS (Heinkel, 2000, Christen, 1997), and MODELICA (Modelica Design Group, 2000) differ in many aspects. This work utilises an aircraft library (Moormann et al., 1999, Moormann, 2001) developed using MODELICA which allows to build domain-specific graphical component libraries and supports many features known from object-oriented programming such as inheritance, packages, etc. For the modelling and the symbolic processing task DYMOLA (Dymola, 2002) was used. It provides a graphical user interface for model composition. The symbolic engine of DYMOLA generates C-code from a MODELICA model. Then a standard C-compiler generates the executable simulation code.

This configuration is already powerful enough to model most parts of the aircraft and to simulate the resulting complex DAE system including certain discontinuities (a so-called 'hybrid DAE'). However, for simulating DAEs with dynamically changing index, the current symbolic engine of DYMOLA (version 4.1d) is too limited. Therefore, a specially developed environment, HYBRISIM (Mosterman and Biswas, 1999), which is based on hybrid bond graphs (Mosterman and Biswas, 1995) was used to model the components with variable index, i.e., the hydraulic actuators. The C-code generated by the two environments, DYMOLA and HYBRISIM

was then merged manually and simulated using a general purpose hybrid dynamic system simulator MASIM (Mosterman, 2001).

The purely discrete-event parts of the elevator control, i.e., the redundancy management, are modelled by a domain-specific formalism including statecharts. It will be shown that the syntax and the semantics of the object-oriented modelling paradigm is not well suited to represent the objects of such a formalism. Instead, a separate modelling environment has been implemented which supports this formalism and generates a monolithic MODELICA component that can be integrated into the MODELICA aircraft model. The behaviour of this component is defined by an algorithm that is interpreted by MODELICA as an additional model constraint, i.e., it is equivalent to one equation with multiple input and output variables. In contrast to the other parts of the model the causality of the interface variables and the calculation scheme of the resulting object are predetermined. While the actuator model is integrated on the data level, the redundancy management model is integrated on the model level.

Section 2 presents a system level view of the elevator redundancy control. Section 3 discusses the different parts in detail and presents the respective models. Simulation results are given in Sect. 4. Finally some conclusions are drawn in Sect. 5.

2 Aircraft Elevator Control System

The aircraft elevator control system includes several forms of redundancy (Seebeck, 1998). The system itself consists of two elevators, the control surfaces. Each of these are controlled by one of two hydraulic actuators while the other one is operating as a passive load. The four actuators take their power from three hydraulic subsystems as depicted in Fig. 2. Two primary flight control units are available to compute actuator control signals and modes.

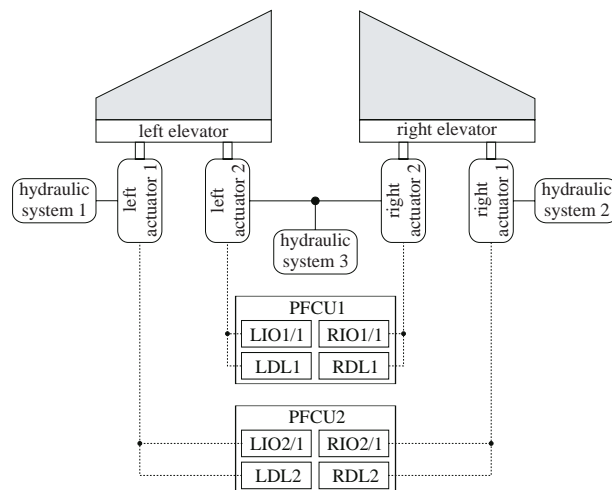


Fig. 2. Elevator system

The functionality of each actuator is specified in textual form in terms of a number of module actuator control modes (MACM) all with their specific behaviour characteristics. These are defined in Table 1. Note that the MACM definitions include behavioural information along with structural information about the particular mode of operation of the actuator components.

Table 1. Module actuator control modes

MACM	Description	Actuator	
		Servo valve	Spool valve
active	The module controls the servo valve in closed loop mode. The corresponding actuator is active and controls the elevator movement.	controlled	open
hot and standby	The module controls the servo valve in closed loop mode. The corresponding actuator is not active and operates as a load.	controlled	closed
passive	The module is waiting and does not generate actuator control signals. <i>It can change its mode at any time to take on control of the corresponding actuator.</i>	not controlled	closed
off	The module is turned off temporarily because of an intermittent failure and does not generate actuator control signals. <i>As long as the failure has not been fixed, it cannot change to a mode where it controls the corresponding actuator.</i>	not controlled	closed
isolated	The module is turned off indefinitely. <i>A persistent fault in the control loop of the corresponding system isolates the module and it cannot change to a mode where it controls the corresponding actuator.</i>	not controlled	closed

The discrete outputs of the redundancy management system are transformed into physical behaviour by means of a spool valve and a servo valve in the hydraulic actuator. Power is supplied by one of the hydraulic systems and delivered to the actuator cylinder that positions the elevator. This flow of energy is modulated by the servo valve, the modulation is computed by a PID feedback control law. The control signals for the actuators are generated by two *primary flight control units* (PFCU) that can operate as *input-output* modules (IOM) or as *direct-link* modules (DLM) controlled by a switch in the control law. The IOMs calculate setpoint values for the actuators based on a PID control algorithm and monitor a number of critical system variables and change between the modes in response. The DLMs allow limited but direct control of the actuators in case the IOMs are not available. The control modules can be in different modes for each of the actuators separately. Moreover, they

may control other aircraft actuators as well. In addition, the servo valve may not be controlled and its piston then is in a default position. Also, the spool valve can be turned on and off to switch between active control and passive loading. Continuous feedback control drives the elevator to its desired setpoint, while higher level redundancy management selects the active actuator and the control law to be used.

Interaction between the actuator and the aircraft model consist of forces and moments acting on the elevator that is stiffly connected to the actuator positioning cylinder as well as the pressure generated by the hydraulic systems. Three hydraulic systems supply the oil for the actuators shown in Fig. 1. When a failure occurs, the redundancy management switches between actuators and oil supply systems to achieve maximum control.

The behavioural redundancy requirements may be formalised by a set of rules for the redundancy management to switch between module actuator control modes as follows (Seebeck, 1998):

1. Mode changes only occur when
 - a system failure is detected, or
 - control of an uncontrolled elevator is requested, or
 - one module requests control of both elevators which are controlled by separate modules.
2. One module should be simultaneously in either *active*, *hot*, or *standby* for both elevators as long as possible.
3. If not overruled by the previous specification, the module priority is such that the switching sequence is IOM2/1 \rightarrow IOM1/1 \rightarrow DLM2 \rightarrow DLM1.
4. There is always one and only one module that controls one elevator, i.e., that is *active*.
5. In case of a failure of the controlling module, control is assumed by a module that is *hot* or *standby*. If no module is in this mode, the one with highest priority that is *passive* assumes control.
6. A module switches to *hot* when the other module that controls the same elevator, and, therefore, is *active*, belongs to another PFCU and both elevators are controlled by IOMs.
7. A module switches to *standby* when the other module that controls the same elevator, and, therefore, is *active*, belongs to another PFCU and one of the elevators is controlled by a DLM.
8. In case of pressure failure, the ‘low pressure’ signal only serves for fault classification. It does not cause a direct mode change.
9. In case of ‘low pressure’ and if a sensor detects an elevator positioning system failure, the module switches *off*. The module switches back to *passive* only when no system failure is reported and the ‘low pressure’ condition does not hold anymore.
10. If ‘low pressure’ is not reported and the elevator positioning system is reported to fail then the module switches to *isolated*.

To prevent nondeterministic switching, priorities are assigned to the possible transitions. Because of the critical nature of switching to the *isolated* mode to prevent

damage to the system, this transition has the highest priority. In addition this causes another module to immediately assume control. This is also desired when, e.g., a pressure loss is detected and the module switches *off*. Therefore, the corresponding transition has second highest priority. Another decision criterion is to allow modules to take over control as quickly as possible. As a result, modes that implement as much control as possible should have highest priority. So, when a module can be switched *active* this should be immediately executed rather than first switching to *standby* if this transition is also enabled. This yields the following priorities:

1. Transition to *isolated*
2. Transition to *off*
3. Transition to *active*
4. Transition to *hot*, *standby*, and *passive*.

Sensors in the elevator control system provide the PFCUs with information about the functioning of the system. In case of abnormal readings, the entire set of measurements is used to infer a particular failure mode. Details of this inference mechanism are beyond the scope of this paper. To test the redundancy management, failure mode effect (FME) analysis investigates the availability of the system for several test cases that embody a set of sensor readings:

- Pressure decrease in the hydraulic system (H1, H2, H3)
- Predefined set of failures (F)
 - IO module failure (1, 2)
 - DL module failure (1, 2)
 - Actuator failure (left inner/outer, right inner/outer).

These failures represent abstractions of actual physical phenomena underlying the failure detection. FME is still the most important step in verifying system safety and reliability of discrete-event control (Mai and Schröder, 1999, Osder, 1999).

The combined discrete redundancy management for two of the four actuators on each of the four modules results in eight redundancy modules. This adds up to a considerable discrete behavioural complexity. Each module consists of six possible local modes and there are eight such modules. Thus, the total number of modes of the redundancy management control is 48. There is always one and only one active state in each of the discrete-event models. But, because of the redundancy specification, each of the models needs to have information about the mode of each of the other ones. This interaction is based on the MACMs and causes logic connections between each of the actuator control modules. Finally, an additional discrete-event model is used to model possible fault scenarios by activating states that correspond to particular failure modes. This model has eleven states.

3 Modelling the Parts of the System

The elevator control system described in Sect. 2 contains a number of parts that are best captured by different modelling approaches: (i) the aircraft dynamics, (ii) the redundancy control, including control law switching, and (iii) the actuator switching behaviour.

3.1 Aircraft Dynamics

To investigate the effect of actuator switching on the overall flight characteristics such as nick rate (q) and angle of attack (α), an aircraft model is required. The more realistic this model, the higher is the probability that the analysis results also hold for the actual implementation on the aircraft.

Object-Oriented Modelling. The design of a realistic aircraft model is a tremendous task that combines several domains within aircraft design such as (i) aerodynamics, (ii) gravity, atmospheric, and wind models, (iii) engine/thrust models, (iv) rigid body models including the effects of fuel consumption, and (v) systems models for primary (attitude) control.

Traditionally, such complex aircraft models are written in a computer processable format such as, e.g., FORTRAN, and they are completely integrated with facilities for behaviour generation, e.g., the numerical solver. This, however, renders the models unwieldy, error-prone, and rather costly to implement and update.

Recently, a more structured approach to aircraft modelling has been developed based on object-oriented modelling techniques and the use of libraries of the domain specific components mentioned before (Moormann et al., 1999, Moormann, 2001). Object-oriented modelling techniques rely on the notion of *encapsulation* to hide the details of physical component models and to increase maintainability. Furthermore, the models are organised hierarchically which allows successive refinement of behaviours at increasing levels of detail.

Graphical Syntax. Figure 3 shows a top-level view of the aircraft model with the engine objects (left), the systems component (top) and the aerodynamics model (right), the rigid body model, and the gravity/atmosphere/wind models (bottom-right). These components can be decomposed hierarchically in similar object diagrams.

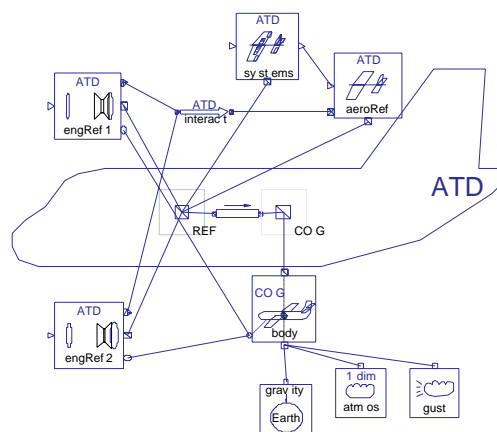


Fig. 3. Top level object diagram of the aircraft model

Communication between objects is realised through ports that also constitute the interface to the next level in the hierarchical decomposition. For a set of connected variables, v_i , these ports use two different connection semantics, (i) $\forall i (i \neq 0 | v_i = v_0)$, i.e., all connected variables are set equal, and (ii) $\sum_i v_i = 0$, i.e., the connected variables are summed to 0. This allows for a convenient implementation of energy flows across ports where the different semantics correspond to the across and through variables, respectively, the product of which constitutes power.

Execution Model. The behaviour of each of the primitive model objects is described in terms of algebraic and differential equations. These are treated as non-causal, i.e., no computational direction of the variables is assigned (it is not determined which variable is to be computed from an equation), which is a convenient way of modelling physical systems in terms of declarative constraint specification. Furthermore, it enhances model reuse.

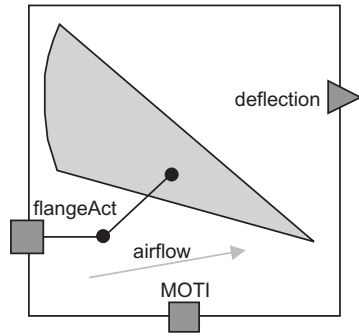


Fig. 4. Elevator control surface library component

To illustrate, consider the elevator control surface library component in Fig. 4. This surface consists of one or more movable parts to adjust the aerodynamic force acting on the aircraft. The library component is connected to the remainder of the aircraft model by three ports: (i) a mechanical port *flangeAct*, that contains the elevator deflection, δ , (ii) an aerodynamic port, *deflection*, that carries the forces because of the airflow around the elevator, and (iii) a mechanical port, *MOTI*, that contains the force acting on the aircraft. Table 2 itemises the most important interface variables of the ControlSurface class. The elevator computes the force F_{act} from

$$0 = f(F_{act}, V_a^2, \delta, \rho, S_{cs}, \dots, m_{cs}, g, \dots), \quad (1)$$

the deflection δ from

$$\delta = k_{kin} x_{act}, \quad (2)$$

and, finally, its rate of change $\dot{\delta}$ from

$$\dot{\delta} = k_{kin} v_{act}. \quad (3)$$

Note that k_{kin} is a parameter internal to the object that represents the kinematics of the mechanism.

Table 2. ControlSurface class interface variables

Interface Variables		
across	x_{act}	displacement of actuator flange
	v_{act}	displacement velocity of actuator flange
	V_a	airspeed velocity
	ρ	air density
	g	gravitational acceleration
through	F_{act}	force acting on actuator flange

To enable the execution, the primitive object equations and the connection constraints for across and through variables are accumulated by a global model interpretation scheme. It sorts and solves the overall system of differential and algebraic (DAE) equations by assigning causality so that the unknowns can be computed from the equations and input and state variables. Algebraic manipulations are performed to reduce the system of equations, e.g., (Andersson, 1994).

To represent switching, equations may be conditionally active. When the conditions change their truth value, this causes *events*. When events occur, variables may undergo discontinuous changes. In addition to the differential and algebraic equations, a ‘pre’ operator is defined to allow access to the value of a variable immediately before a discontinuous change. Because this introduces discrete state behaviour, an iteration is required to converge to a consistent state before the continuous simulation is resumed. Though this mechanism can be used for implementing discrete-event behaviour, it is difficult to mimic state transition diagrams using object diagrams and even more so to describe the state transition behaviour by local equations of the primitive states and transitions. The graphical syntax of object diagrams does not allow annotation of component connections, thus it is not possible to write conditions, events, and actions alongside a transition. Furthermore, transitions are not objects in object diagrams. Therefore, the transition behaviour requires a specific transition object to be inserted. Execution has to be described in terms of local algebraic constraints that communicate between states and transitions to evaluate whether a state is active and a transition is enabled (Mosterman et al., 1998).

The result of collecting the local equations, adding the connection constraints, and sorting and solving these leads to a global system of equations of the form

$$\begin{aligned} \dot{x} &= f_{\alpha}(x, u, t) \\ 0 &= g_{\alpha}(x, u, t) \\ \alpha^+ &= \phi_{\alpha}(x, u, t) \end{aligned} \tag{4}$$

where f_α specifies the dynamics in mode α , g_α the event generation functions ('zero crossings'), and ϕ_α the next mode function. Before continuous simulation can start or be resumed after an event occurred, a consistent mode α , i.e., $\alpha^+ = \alpha$, has to be found. Typically, this is performed by a fixed point iteration scheme.

3.2 Redundancy and Position Control

The main purpose of the two primary flight control units is the generation of appropriate continuous and discrete control signals for the four elevator actuators. Each PFCU contains a failure monitoring function, a specific discrete-event part for the redundancy management as well as one feedback and direct link controller per elevator. When a failure is detected, the redundancy management parts of both PFCUs interact tightly in order to achieve a consistent decision on the appropriate reaction, before switching the operating control laws. This is because each PFCU is responsible for different actuators and has to take the discrete state of the other PFCU into account in order to guarantee that each elevator is controlled by one actuator only. Therefore a simple failure may trigger a sequence of transitions in both PFCUs, where a discrete mode transition in one PFCU may lead to a state which forces another transition in the other PFCU and so on.

Graphics. Since hardware aspects are beyond the scope of this paper, the redundancy management parts of both PFCUs are unified in one discrete-event model component neglecting the distributed architecture of the system. As a consequence, the aircraft model contains only one elevator control component. This is divided into three parts: (i) a failure injection module that replaces the failure monitoring functions so that specific failure scenarios can be studied, (ii) the combined redundancy management parts of both PFCUs that react on changes of the failure configuration and (iii) the switched position controllers of both PFCUs the transfer functions of which depend on the actual modes of the redundancy management component (Fig. 5).

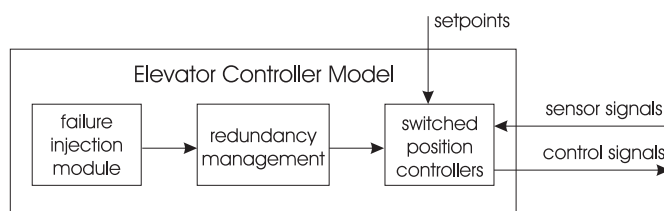


Fig. 5. The structure of the elevator controller model

The requirements for the redundancy management which were formulated informally in Section 2 state that each redundancy module contains 6 possible local

modes. Since a redundancy module switches from one mode to another under certain conditions, the modules should be modelled by a kind of state transition diagram, where the modes are represented by discrete states and the transition arrows represent the possible mode switchings. In order to take the transition priorities into consideration, hierarchical states as known from the statechart formalism (Harel, 1987) are used. Additional states are introduced that do not correspond to a mode, but represent the priorities of the transitions: The higher a state in the hierarchy the higher is the priority of its outgoing transitions, e.g., the transition ToOff in Fig. 6 has a higher priority than ToAct and a lower priority than ToIso. The statechart model in Fig. 6 reflects the state transition aspects of a redundancy module declared in the informal description of the requirements.

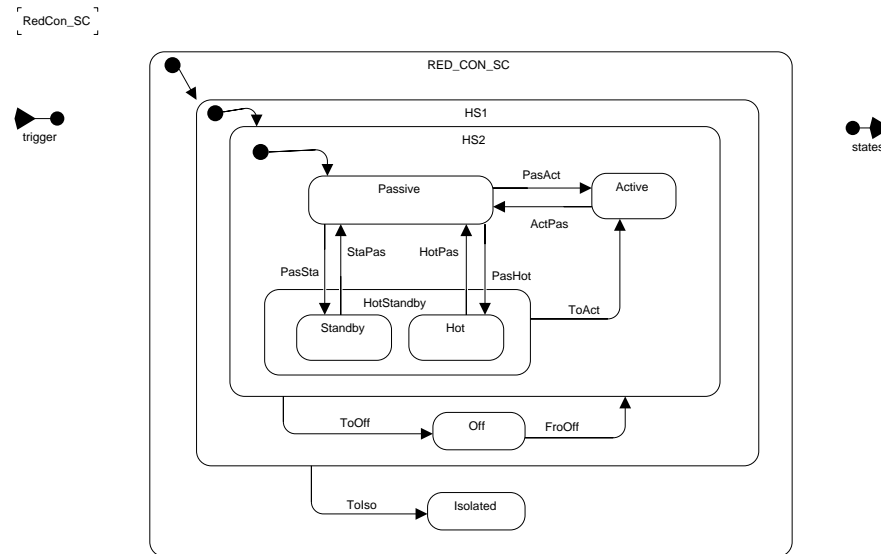


Fig. 6. The redundancy module statechart

The transition conditions can be derived from the switching rules of the requirements and differ for each statechart. In order to keep the statechart model generic and take architectural aspects into consideration, a specific hierarchical block diagram formalism is used (Fig. 7). Two blocks on the top level represent the two primary flight control units. The input port contains the failure values that originate from the failure injection module whereas the output ports transmit the actual module modes to the switched controllers. Each PFCU block contains four control modules (LIO, RIO, LDL and RDL) as subblocks. Their behaviour is defined by the statechart in Fig. 6. The transition conditions are calculated outside the statecharts in a special block (PFCU1_Logic and PFCU2_Logic) with no state behaviour.

Execution The intended behaviour of the elevator control model is as follows: The failure injection module generates Boolean signals that indicate the presence

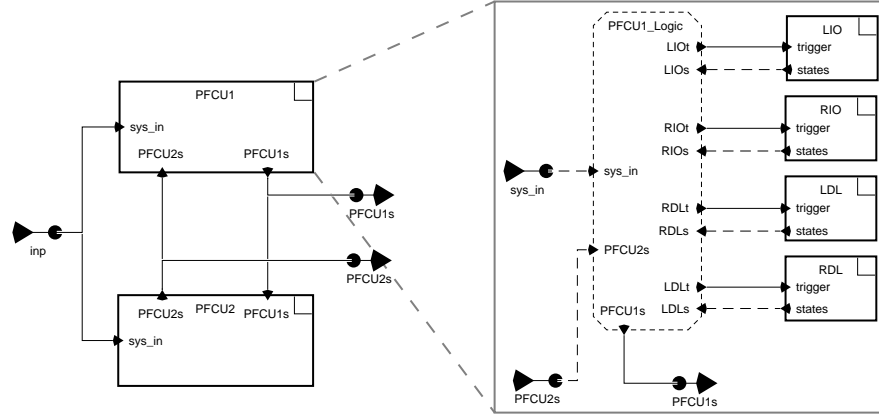


Fig. 7. Two block diagrams of the discrete-event part

of specific failures. When a failure signal changes, the transition conditions of both PFCUs are evaluated and their values are transmitted to the statechart blocks that perform their transitions independently. After all statecharts have converged to a persistent discrete state, i.e. no further transitions happen, the transition conditions are calculated again taking the new states into account and a new set of transitions may be performed in the modules again. When the overall discrete-event system reaches a stable state, this local event iteration is stopped, the output values are set, and the position controllers may change their mode.

In order to analyse the behaviour of the elevator control and the overall aircraft for different failure scenarios, the failure injection module generates predetermined sequences of failures. These scenarios can be modelled by equations containing logical expressions and inequalities over the independent variable time and parameters as shown in the following example where IO2failure is present from time t_1 to t_2 :

$$\text{IO2failure} = (t > t_1) \wedge (t < t_2). \quad (5)$$

The output of the redundancy management part switches the position controllers that are easily described using equations. The following example shows the controller equations of PFCU1 for the left elevator:

$$e_{act,l1} = w_{act} - x_{act,l} \quad (6)$$

$$u_{act} = \begin{cases} 0 & \text{if PFCU1states.LIO is Off or Isolated} \\ w_{act} & \text{else if PFCU1states.LDL is Active,} \\ k_p e_{act,l1} + k_d v_{act,l} & \text{else} \end{cases} \quad (7)$$

$$u_{spool,l1} = \text{PFCU1states.LIO.Active} \quad (8)$$

3.3 Actuator Dynamics

The hydraulic actuators are the interface between the discrete-event domain of redundancy control and the continuous domain of the aircraft dynamics. The actuator

here is not modelled with all details as this would lead to steep gradients in the behaviour that are difficult to handle and slow down simulation of the aircraft behaviour, even if efficient numerical solvers such as DASSL (Petzold, 1982) are used.

Higher Index DAE. The decision to remove small physical effects such as fluid storage in lines and oil elasticity and viscosity leads to DAEs with a higher complexity because state variables are then directly coupled instead of interacting through additional states with small time constants. These DAEs can be transformed by differentiation before simulation run, but the switching effects of the actuators may also cause such algebraic constraints to emerge during simulation, requiring two phenomena to be handled: (i) the state variables that become algebraically coupled are constrained to a subspace of reduced dimension and the values before the constraint becomes active have to be projected into this subspace, and (ii) the future dynamic behaviour of these state variables must be in this reduced subspace.

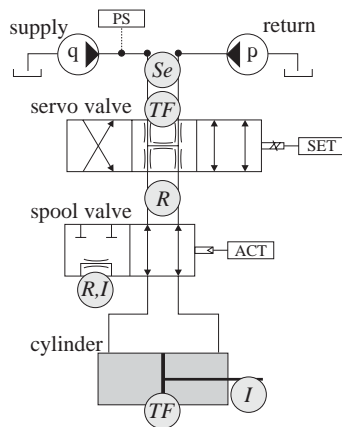


Fig. 8. Schematic of hydraulic actuator

To illustrate these effects, consider the actuator model in Fig. 8. When initially the actuator is *active*, the supply path is open, i.e., control signals generated by the servo valve are supplied to the positioning cylinder, causing the piston to accelerate. When, at a given point in time, the actuator is switched to be *off*, the loading path becomes active. Because of the inertial effects in the loading pathway, there is dependency between the piston and this fluid inertia and an algebraic constraint between these two variables ($v_{piston} = -A_p f_{load}$) restricts the state space in which the system evolves. This is illustrated in Fig. 9(a), where the double arrow heads on the dashed field lines indicate the direction of the discontinuous change. This algebraic dependency would be eliminated by introducing small parasitic storage effects for the piping and some oil elasticity and viscosity, but this adds very steep gradients to overall system behaviour as illustrated by Fig. 9(b) that complicate simulation and are not relevant for the overall behaviour of the aircraft.

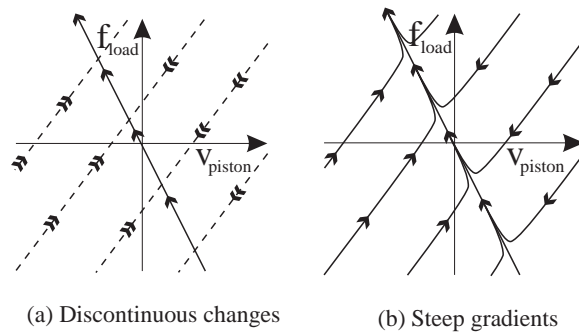


Fig. 9. Phase space for v_p and f_{load}

The implicit jumps in the state variable values have to be computed during simulation. At present, commercially available simulation tools cannot handle such abrupt changes in DAE models. Therefore the experimental modelling and simulation environment HYBRSIM (Mosterman and Biswas, 1999) was used which has been realised for the purpose of testing algorithms for the reinitialisation of switched systems with index changes. HYBRSIM is based on bond graph modelling of the physical system.

Bond Graph Model of the Actuators. Figure 10 shows the hybrid bond graph model of the two left hydraulic actuators. The two *Se* elements¹ are sources (inputs) of a bond graph model which are connected to the hydraulic circuits in the aircraft model that provide the input pressure. The servo valve modulation is applied by the *TF* elements, where the *setL1* and *setL2* elements are connected to the setpoint generated by the aircraft control model. The *I* elements represent connections (equal flow points) and the attached *R* element captures dissipative effects. Note that these are modelled as linear phenomena. The *loadL1* (*loadL2*) connection also has some inertia associated with it, embodied by the *IloadL1* (*IloadL2*) element. The cylinder chamber is modelled by a *0* element, an equal pressure point. Both cylinders connect through a piston with area modelled by a *TF* element to one equal velocity point for the elevator control surface movement. This velocity, as well as the displacement and force are inputs to the aircraft model.

The switching behaviour is modelled by two *controlled junctions* (Mosterman and Biswas, 1995) in each actuator, in the left actuator these are *supplyL1* and *loadL1*. The local finite state machines that control their states are given in Fig. 11. The control event *actL1* is generated by the redundancy control in the enclosing part of the model. When the *supplyL1* junction is ON and *loadL1* is OFF, the actuator is active. When *supplyL1* is OFF and *loadL1* is ON, it is loading (either hot, standby, passive, or isolated). Note that the mutual switching constraints allow no other configurations.

¹ The element type is listed on the left of each element rectangle.

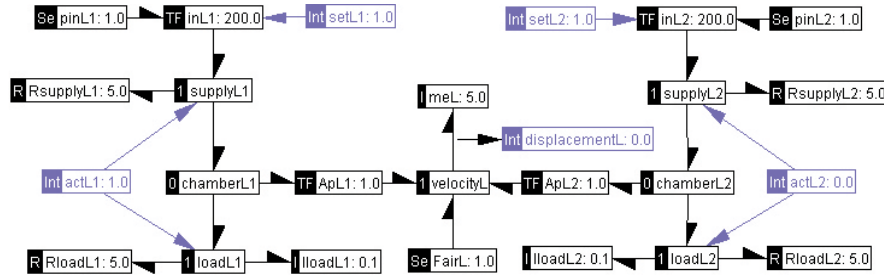


Fig. 10. Hybrid bond graph of the two left hydraulic actuators

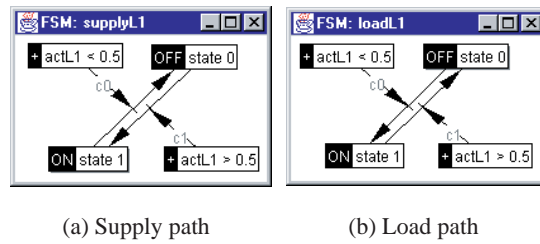


Fig. 11. Finite state machines of actuator 1 in the hybrid bond graph

Equations. The equations generated from the hybrid bond graph by HYBRSIM incorporate the switching effect as guarded equations. This prevents the need for pre-enumeration which would cause an exponential growth of the number of modes.² For example, for the loading pathway, *loadL1*, the equation generated is

$$0 = (-chamberL1.p + lloadL1.p + RloadL1.p)\alpha_i + (loadL1.f) \cdot (1 - \alpha_i) \quad (9)$$

where α_i is the i^{th} entry in the mode vector α . This ensures that in a mode where this connection is active, $\alpha_i = 1$, the pressure drops of the connected elements are balanced. When the connector is not active, $\alpha_i = 0$, the fluid flow through *loadL1* becomes 0. This models ideal switching but may lead to higher index DAEs (e.g., because *lloadL1* and *mpL* become algebraically related). A numerical solver such as DASSL can handle systems up to index 1 directly and up to index 2 with some provisions, e.g., the step-size control of index 2 variables needs to be switched off (Bujakiewicz, 1994). Another prerequisite is that DASSL should be given a set of consistent initial conditions, i.e., those that are in the correct subspace of continuous behaviours. This is achieved by applying a projection mechanism which is consistent with physical conservation laws (Griepentrog and März, 1986, van der Schaft and Schumacher, 1996, Verghese et al., 1981).

² For the hybrid bond graph in Fig. 10 there are already $2^4 = 16$ possible modes, but only two occur during normal operation.

The discontinuous changes are computed by first linearising the system with a finite difference method. Then a pseudo Weierstrass normal form is derived (up till index 2)

$$0 = \begin{bmatrix} \bar{E}_{11} & 0 & 0 \\ 0 & 0 & \bar{E}_{22,12} \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \dot{\bar{x}}_1 \\ \dot{\bar{x}}_{2,1} \\ \dot{\bar{x}}_{2,2} \end{bmatrix} + \begin{bmatrix} \bar{A}_{11} & \bar{A}_{12,1} & \bar{A}_{12,2} \\ 0 & \bar{A}_{22,11} & \bar{A}_{22,12} \\ 0 & 0 & \bar{A}_{22,22} \end{bmatrix} \begin{bmatrix} \bar{x}_1 \\ \bar{x}_{2,1} \\ \bar{x}_{2,2} \end{bmatrix} + \begin{bmatrix} \bar{B}_1 \\ \bar{B}_{2,1} \\ \bar{B}_{2,2} \end{bmatrix} [u], \quad (10)$$

where $\bar{E}_{11,11}$, $\bar{A}_{22,11}$, and $\bar{A}_{22,22}$ are of full rank. This allows computation of the initial conditions as (Mosterman, 2000b)

$$\begin{aligned} \bar{x}_1 &= \bar{x}_1^0 + \bar{E}_{11}^{-1} \bar{A}_{12,1} \bar{A}_{22,11}^{-1} \bar{E}_{22,12} (\bar{x}_{2,2} - \bar{x}_{2,2}^0) \\ \bar{x}_{2,1} &= -\bar{A}_{22,11}^{-1} (\bar{B}_{2,1} u + \bar{E}_{22,12} \dot{\bar{x}}_{2,2} + \bar{A}_{22,12} \bar{x}_{2,2}) \\ \bar{x}_{2,2} &= -\bar{A}_{22,22}^{-1} \bar{B}_{2,2} u, \end{aligned} \quad (11)$$

where \bar{x}_0 are the user-provided initial values after the coordinate transformation to achieve the desired normal form, $\bar{x}_0 = Zx_0$. The values for \bar{x} can then be transformed back to obtain initial values for x that are in the correct subspace of the dynamic behaviour, and in this manner the implicit jump is determined.

4 Simulation of the Overall System

The aircraft model, the redundancy control system, and the actuator feedback and discrete event control were modelled using different modelling formalisms and tools (DYMOLA, HYBRISIM, DOME). Each of these is best suited for the respective task. To enable a comprehensive analysis, however, the parts have to be integrated into a coherent model.

4.1 Integrating the Components

Since the descriptions of the failure injection module and the redundancy management system laws are based on equations, they can be incorporated easily into the object-oriented and equation-based aircraft model. This also holds for the hydraulic actuators, in principle, because the bond graph models correspond to a set of hybrid differential and algebraic equations. But due to present restrictions of the simulation software available for object-oriented modelling languages, specific simulation code is generated from the bond graphs of the actuators and merged with the C-code that results from the aircraft model.

For the redundancy management component, the modelling environment generates a simulation algorithm that defines the input-output behaviour of the discrete-event component. This automatically generated algorithm is designed in a way that is compatible to the MODELICA language so that it can be embedded directly into the aircraft model. In MODELICA such an algorithm is regarded simply as an additional model constraint that corresponds to an equation that contains a function with a fixed set of input and output variables.

To simulate the resulting hybrid model, MODELICA's hybrid DAE semantics is exploited. The temporal inequality expressions in the failure injection module are transformed into time events for the numerical integrator so that the continuous integration stops exactly when a switching time has elapsed. Then the whole set of equations is re-evaluated with the new values of the inequality expressions. Thereby, the algorithm of the redundancy management is also re-evaluated resulting possibly in a new state which may switch the feedback control laws.

4.2 Simulation Results

The phugoid in Fig. 12 is the result of two interacting phenomena: When the aircraft pitch angle increases, it gains altitude and at the same time loses airspeed. Because of this loss of airspeed, there is less upward thrust, which causes the aircraft to lose altitude in return. However, as it starts losing altitude, it picks up speed again and the airspeed rises. This results in a slightly damped oscillatory behaviour which is required to be stable in commercial aircrafts.

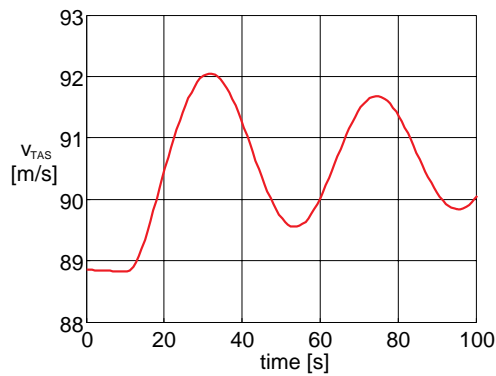


Fig. 12. Simulation shows a phugoid typical for aircraft

To investigate the effect of the redundancy control on the aircraft's behaviour, an actuator failure is introduced during a setpoint change. The setpoint change occurs at $t = 0.05$ [s] and the actuator failure at $t = 0.08$ [s]. Figure 13 shows that the failure leads to an immediate change of the active actuators and the switching transients in the hydraulics cause a sharp drop in elevator velocity. Because small effects such as oil elasticity and viscosity are neglected in the simulation, this results in a discontinuous change that occurs because of the algebraic dependency between elevator inertia and fluid inertia of the new loading path.

During a short period of time, the PID control causes the elevator velocity to ramp up to the value which it would have assumed without the failure. Note the short delay that is possible because the actuator that switches to active was *hot* and *shadowing* the PID control.

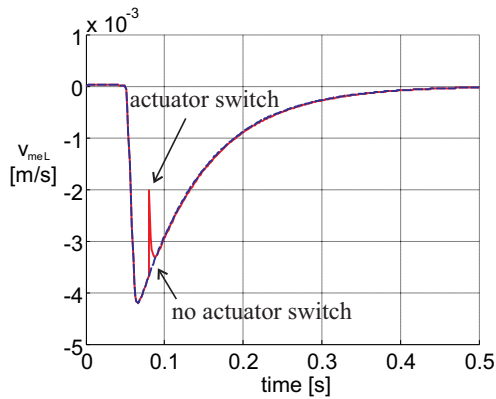


Fig. 13. Elevator velocity when a failure occurs at $t = 0.08$ shortly after a setpoint change at $t = 0.05$

The aircraft redundancy control is designed such that an actuator failure should not have a noticeable effect on the behaviour of the aircraft. Using the comprehensive model with switching logic and transients, and an extensive model of the aircraft dynamics, this effect can be studied as well. Figure 14(b) shows the effect of the actuator switch on the aircraft pitch angle, and Fig. 15(b) shows the effect on the pitch angle velocity. This verifies that the actuator switch has almost no effect on the overall aircraft behaviour which, because of the realistic aircraft model, provides much confidence for the real implementation. Note that the small effect of the actuator switching on the global behaviour manifests itself after a significant delay.

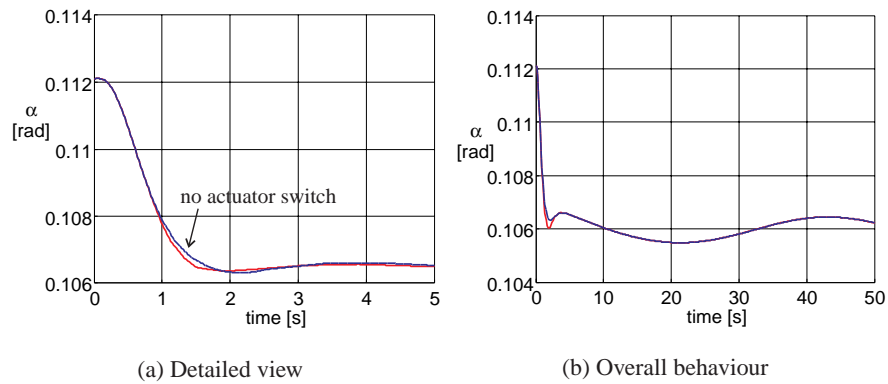


Fig. 14. Pitch angle for normal behaviour and for an actuator switch at $t = 0.08$

Table 3 illustrates how the redundancy management reacts, when the IO module failure occurs in PFCU2. In this case, all resulting state transitions are symmetrical, i.e., the modules of the right elevator have always the same state as the corresponding modules of the left elevator. Therefore the given states refer to both sides. In

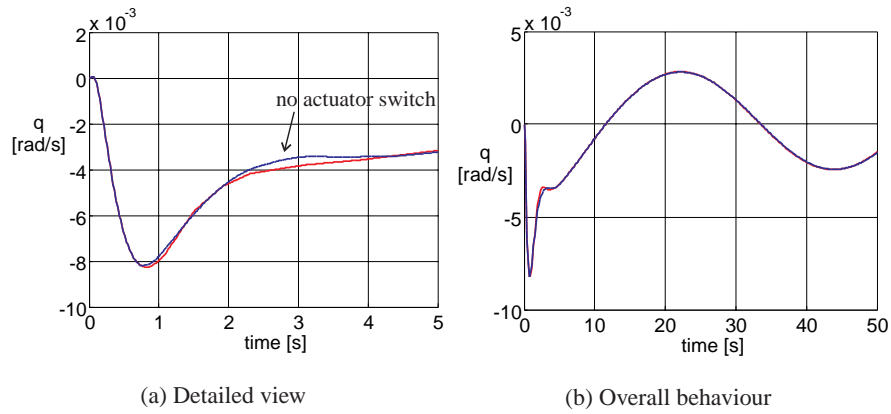


Fig. 15. Pitch angle velocity for normal behaviour and for an actuator switch at $t = 0.08$

the first local transition the statecharts of LIO and RIO (Left / Right IO) of PFCU2 switch from *Active* to *Isolated*, since these modules should not be activated again (see rules 1 and 10 in Section 2). Then PFCU1 takes over the actuators by activating its LIO and RIO modules (rules 1, 3, 5). In the last local transition, the LDL and RDL (Left / Right DL) statecharts of PFCU2 switch into the *Hot* mode preparing the system for a possible second failure (rule 6). Since state 2 would violate rule 4 and the transition from state 3 to state 4 would violate rule 1, the internal iterations have to be hidden from the outer system in order to prevent inconsistent outputs. This is why only the global transition from state 1 to state 4 is made observable to the outside.

Table 3. State transitions of the redundancy management system

components		local steps			
		1	2	3	4
PFCU2	RIO/LIO	Active	Isolated	Isolated	Isolated
	RDL/LDL	Passive	Passive	Passive	Hot
PFCU1	RIO/LIO	Hot	Hot	Active	Active
	RDL/LDL	Passive	Passive	Passive	Passive
outer actuators		control	—	—	shadow
inner actuators		shadow	—	—	control
global visibility		yes	no	no	yes

5 Conclusions

The comprehensive model of the aircraft developed here incorporates the redundancy management system, the switched positioning controllers, the actuator models as well as a complex model of the general dynamics of the aircraft. Hence, it is possible to assess the design of the elevator control system with respect to the overall behaviour of the aircraft in the case of failures. Since the less important physical effects of the hydraulic actuators were neglected, the simulation is fast enough to be used also in the context of a multi-objective parameter optimisation (MOPS) (Joos, 1999). Such an optimisation may, e.g., reduce the elevator surface or the actuator power such that the switching transients still do not affect the level of aircraft handling.

The abstractions used in the actuator models, i.e. neglecting small physical effects such as oil elasticity and viscosity, result in a DAE that may change its index during simulation. A standard DAE solver, such as DASSL, can be applied for this model, if the re-initialisation at event times results in a consistent state. For a correct behavioural simulation, this re-initialisation has to satisfy the physical conservation laws. For the purpose of this feasibility study the actuators were modelled in HYBR-SIM, a modelling environment based on hybrid bond graphs that supports the necessary re-initialisation procedure. The C-code generated by this environment was manually combined with the C-code generated by DYMOLA which includes the rest of the aircraft model. The hybrid system simulator MASIM was used to generate behaviors. MASIM has facilities to compute discontinuous changes of generalized state variables as algebraic constraints between them become active. The discrete-event parts of the aircraft are modelled using a visual specification language and are translated into a MODELICA algorithm that can be integrated into the aircraft model on the model level (Mosterman et al., 2002).

The presented modelling and simulation approach that combines an object-oriented modelling language such as MODELICA, domain-specific model libraries, discrete-event modelling formalisms and powerful simulation methods including correct state re-initialisation, was successfully applied to the aircraft elevator control system and seems to be promising for general complex technological systems.

References

- ABACUSS (1995). <http://yoric.mit.edu/abacuss/abacuss.html>. Massachusetts Institute of Technology.
- Abadi, M. and Cardelli, L. (1996). *A Theory of Objects*. Springer, New York.
- Abel, D. (1990). *Petri-Netze für Ingenieure*. Springer, Berlin, Germany.
- Adjiman, C., Schweiger, C., and Floudas, C. (1998). Mixed-integer nonlinear optimization in process synthesis. In Du, D.-Z. and Pardalos, P., editors, *Handbook of Combinatorial Optimization*, volume 1, pages 1–76. Kluwer Academic Publisher.
- Albro, J. and Bobrow, J. (2001). Optimal motion primitives for a 5 DOF experimental hopper. In *Proceedings of the IEEE International Conference on Robotics and Automation (Seoul, Korea)*, pages 3630–3635.
- Allgor, R. and Barton, P. (1997). Mixed integer dynamic optimization. *Computational Chemical Engineering*, 21:451–456.
- Alur, R., Courcoubetis, C., Halbwachs, N., Henzinger, T., Ho, P.-H., Nicollin, X., Olivero, A., Sifakis, J., and Yovine, S. (1995). The algorithmic analysis of hybrid systems. *Theoretical Computer Science*, 138:3–34.
- Alur, R., Courcoubetis, C., Henzinger, T. A., and Ho, P. H. (1993). Hybrid Automata: An Algorithmic Approach to the Specification and Verification of Hybrid Systems. In Grossmann, R. L., Nerode, A., Ravn, A. P., and Rischel, H., editors, *Hybrid Systems*, volume 736 of *Lecture Notes in Computer Science*, pages 209–229. Springer.
- Alur, R., Dang, T., Esposito, J., Fierro, R., Hur, Y., Ivančić, F., Kumar, V., Lee, I., Mishra, P., Pappas, G., and Sokolsky, O. (2001). Hierarchical hybrid modeling of embedded systems. In Henzinger, T. and Kirsch, C., editors, *EMSOFT 2001: First International Workshop on Embedded Software, Tahoe City, CA, USA, October 8–10, 2001*, volume 2211 of *Lecture Notes in Computer Science*, pages 14–31. Springer.
- Alur, R. and Dill, D. (1990). A theory of timed automata. *Theoretical Computer Science*, 126:183–235.
- Alur, R., Grosu, R., Hur, Y., Kumar, V., and Lee, I. (2000a). Modular specification of hybrid systems in Charon. In *Proc. HSCC’00*, Springer LNCS 1790.
- Alur, R., Henzinger, T., Lafferiere, G., and Pappas, G. (2000b). Discrete abstractions of hybrid systems. *Proceedings of the IEEE*, 88(7):971–984.
- Alur, R. and Henzinger, T. A. (1999). Reactive modules. *Formal Methods in System Design: An International Journal*, 15(1):7–48.
- Alur, R., Henzinger, T. A., and Sontag, E. D., editors (1996). *Hybrid Systems III: Verification and Control*, volume 1066 of *Lecture Notes in Computer Science*. Springer.
- Andersson, M. (1994). *Object-Oriented Modeling and Simulation of Hybrid Systems*. PhD dissertation, Department of Automatic Control, Lund Institute of Technology, Lund, Sweden.

- Antsaklis, P., editor (2000). *Special Issue on Hybrid Systems: Theory and Applications*, volume 88, no. 7 of *Proceedings of the IEEE*.
- Antsaklis, P., Kohn, W., Lemmon, M., Nerode, A., and Sastry, S., editors (1999). *Hybrid Systems V*, volume 1567 of *Lecture Notes in Computer Science*. Springer.
- Antsaklis, P. and Koutsoukos, X. D. (1998). On Hybrid Control of Complex Systems: A Survey. In *Proceedings Hybrid Dynamical Systems*, ADPM '98, pages 1–8, Reims, France.
- Antsaklis, P. and Nerode, A., editors (1998a). *Special Issue on Hybrid Control Systems*, volume 43 of *IEEE Transactions on Automatic Control*.
- Antsaklis, P., Nerode, A., Kohn, W., and Sastry, S., editors (1995). *Hybrid Systems II*, volume 999 of *Lecture Notes in Computer Science*. Springer.
- Antsaklis, P., Nerode, A., Kohn, W., and Sastry, S., editors (1997). *Hybrid Systems IV*, volume 1273 of *Lecture Notes in Computer Science*. Springer.
- Antsaklis, P. J. and Nerode, A. (1998b). Special issue on hybrid systems. *IEEE Transactions on Automatic Control*, 43.
- Apt, K. R., Francez, N., and de Roever, W.-P. (1980). A proof system for communicating sequential processes. *ACM Transactions on Programming Languages and Systems*, 2(3):359–385.
- Asarin, E., Bournez, O., Dang, T., and Maler, O. (2000a). Reachability analysis of piecewise-linear dynamical systems. In *3rd Int. Workshop of Hybrid Systems: Comp. and Control*, volume 1790 of *LNCS*, pages 20–31. Springer.
- Asarin, E., Bournez, O., Dang, T., Maler, O., and Pnueli, A. (2000b). Effective synthesis of switching controllers for linear systems. *Proceedings of the IEEE*, 88:1011–1025.
- Automatica 35(3) (1999). A special issue on hybrid systems. *Automatica*, 35:347–519.
- Back, A., Guckenheimer, J., and Myers, M. (1993). A dynamical simulation facility for hybrid systems. In Grossmann, R., Nerode, A., Ravn, A., and Rischel, H., editors, *Lecture Notes in Computer Science: Hybrid Systems*, volume 736, pages 255–267. Springer.
- Balas, E. (1985). Disjunctive programming and a hierarchy of relaxations for discrete optimization problems. *SIAM Journal Alg. Disc. Meth.*, 6(3):466–486.
- Barros, F. J. (1996). The dynamic structure discrete event system specification formalism. *Transactions of the SCS International*, 13(1):35–46.
- Barton, P. I. (1992). *The Modelling and Simulation of Combined Discrete/Continuous Processes*. PhD dissertation, University of London.
- Bastide, R. (1995). Approaches in unifying Petri nets and the Object-Oriented Approach. In *Object-Oriented Programming and Models of Concurrency 16th International Conference on Application and Theory of Petri Nets*, Italy.
- Baumgarten, B. (1990). *Petri-Netze: Grundlagen und Anwendungen*. BI-Wissenschaftsverlag, Mannheim, Wien, Zürich.
- Bellman, R. (1957). *Dynamic Programming*. Princeton University Press.
- Bemporad, A., Borelli, F., and Morari, M. (2002). On the optimal control law for linear discrete time hybrid systems. In *Hybrid Systems: Computation and Control*, volume 2289 of *LNCS*, pages 105–119. Springer.

- Bemporad, A., Mignone, D., and Morari, M. (1999). An efficient branch and bound algorithm for state estimation and control of hybrid systems. In *Proc. 5th European Control Conference*.
- Bemporad, A. and Morari, M. (1999a). Control of systems integrating logic, dynamics, and constraints. *automatica*, 35(3):407–427.
- Bemporad, A. and Morari, M. (1999b). Verification of hybrid systems using mathematical programming. In Vaandrager, F. W. and van Schuppen, J. H., editors, *Hybrid Systems: Computation and Control, Proc. 2nd Int. Workshop, HSCC'99, Berg en Dal, The Netherlands, March 1999*, Lecture Notes in Computer Science 1569, pages 31–45. Springer.
- Bender, K. and Kaiser, O. (1995). Simultaneous Engineering durch Maschinenemulation. *CIM Management*, 11(4):14–18.
- Benedetto, M. D. D. and Sangiovanni-Vincentelli, A. L., editors (2001). *Hybrid Systems: Computation and Control*, volume 2034 of *Lecture Notes in Computer Science*. Springer.
- Bergstra, J. and Klop, J. (1984). Process algebra for synchronous communication. *Information and Control*, 60(1):109–137.
- Betts, J. (1998). Survey of numerical methods for trajectory optimization. *AIAA Journal of Guidance, Control, and Dynamics*, 21(2):193–207.
- Bhat, G., Cleaveland, R., and Grumberg, O. (1995). Efficient on-the-fly model checking for CTL*. In *LICS '95: 10th Annual IEEE Symposium on Logic in Computer Science, San Diego, California, USA, June 26–29, 1995*, pages 388–397. IEEE Computer Society Press.
- Blanke, M., Frei, C., Kraus, F., Patton, R., and Staroswiecki, M. (2000a). Fault-tolerant control systems. In Isidori, A., Aström, K. J., Blanke, M., Schaufelberger, W., Albertos, P., and Sanz, R., editors, *Control of Complex Systems*, chapter 8, pages 165–189. Springer.
- Blanke, M., Frei, C. W., Kraus, F., Patton, R. J., and Staroswiecki, M. (2000b). What is fault-tolerant control? In *Proceeding of SAFEPROCESS 2000: 4th Symposium on Fault Detection*, page 40. IFAC.
- Bobbio, A., Garg, S., Gribaudo, M., Horváth, A., Sereno, M., and Telek, M. (1999). Modeling software systems with rejuvenation, restoration and checkpointing through fluid stochastic petri nets. In *Proc. Eighth International Workshop on Petri Nets and Performance Models - PNPM'99*, pages 82–91.
- Bolognesi, T. and Brinksma, E. (1987). Introduction to the ISO specification language LOTOS. *Computer Networks*, 14:25–59.
- Brack, G. (1974). *Dynamik technischer Systeme*. VEB Deutscher Verlag für Grundstoffindustrie, Leipzig.
- Branicky, M. (1993). Topology of hybrid systems. In *Proceedings of the 32nd IEEE Conference on Decision and Control (San Antonio, TX)*, pages 2309–2314.
- Branicky, M. (1994a). Analyzing continuous switching systems: Theory and examples. In *Proceedings of the American Control Conference (Baltimore, MD)*, pages 3110–3114.
- Branicky, M. (1994b). Stability of switched and hybrid systems. In *Proceedings of the 33rd IEEE Conference on Decision and Control (Lake Buena Vista, FL)*, pages 3498–3503.

- Branicky, M. (1994c). A unified framework for hybrid control. In *Proceedings of the 33rd IEEE Conference on Decision and Control (Lake Buena Vista, FL)*, pages 4228–4234.
- Branicky, M. (1995). *Studies in Hybrid Systems: Modeling, Analysis and Control*. PhD thesis, Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science.
- Branicky, M. (1996). General hybrid dynamical systems: Modeling, analysis, and control. In Alur, R., Henzinger, T., and Sontag, E., editors, *Lecture Notes in Computer Science: Hybrid Systems III*, volume 1066, pages 186–200. Springer.
- Branicky, M. (1998). Multiple Lyapunov Functions and Other Analysis Tools for Switched and Hybrid Systems. *IEEE Trans. Aut. Control*, 43(4):475–482.
- Branicky, M., Borkar, V., and Mitter, S. (1998). A unified framework for hybrid control: Model and optimal control theory. *IEEE Transactions on Automatic Control*, 43(1):31–45.
- Branicky, M., Hebbbar, R., and Zhang, G. (1999). A fast marching algorithm for hybrid systems. In *Proceedings of the 38th IEEE Conference on Decision and Control (Phoenix, AZ)*, pages 4897–4902.
- Brenan, K. E. and Campbell, S. L. (1996). *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations*. siam.
- Brockett, R. (1993). Hybrid models for motion control systems. In Trentelmann, H. and Willems, J., editors, *Essays on Control: Perspectives in the Theory and its Applications*, pages 29–53. Boston: Birkhäuser.
- Broenink, J., Hilderink, G., and Bakkers, A. (1998). Conceptual design for controller software of mechatronic systems. In Bradshaw, A. and Counsel, J., editors, *Computer aided Conceptual Design '98*.
- Bröhl, A. and Dröschel, W. (1995). *Das V-Modell*. Oldenburg.
- Brooke, A., Kendrick, D., Meeraus, A., and Raman, R. (1998). *GAMS/CPLEX – A User's Guide*. GAMS Development Corporation.
- Brookes, S., Hoare, C., and Roscoe, A. (1984). A theory of communicating sequential processes. *Communications of the ACM*, 31(3):560–599.
- Broucke, M., Di Benedetto, M., Di Gennaro, S., and Sangiovanni-Vincentelli, A. (2000). Theory of optimal control using bisimulations. In *Proc. 3rd Int. Workshop of Hybrid Systems: Comp. and Control*, volume 1790 of *LNCS*, pages 89–102. Springer.
- Brown, J. S. and de Kleer, J. (1990). A qualitative physics based on confluences. In *Qualitative Reasoning about Physical Systems*, pages 88–126. Morgan Kaufmann Publishers, San Mateo, CA.
- Broy, M. (2001). Refinement of time. *Theoretical Computer Science*, 253(1):3–26.
- Bryant, R. E. (1986). Graph-based algorithms for Boolean function manipulation. *IEEE Transactions on Computers*, C-35(8):677–691.
- Bryant, R. E. (1992). Symbolic Boolean manipulation with ordered binary-decision diagrams. *ACM Computing Surveys*, 24(3):293–318. Preprint version published as CMU Technical Report CMU-CS-92-160.
- Buchholz, J. J. (1999). Systemsimulation. Vorlesungsmanuskript.

- Bühler, M. and Koditschek, D. (1993). From stable to chaotic juggling: Theory, simulation, and experiments. In Spong, M., Lewis, F., and Abdallah, C., editors, *Robot Control – Dynamics, Motion Planning, and Analysis*, pages 525–530. New York: IEEE Press.
- Bujakiewicz, P. (1994). *Maximum weighted matching for high index differential algebraic equations*. PhD dissertation, TU Delft, Delft, Netherlands. ISBN 90-9007240-3.
- Buss, M. (1998). Multi-fingered Regrasping using a Hybrid Systems Approach. In *Proceedings of the 2nd IMACS/IEEE International Multiconference on Computational Engineering in Systems Applications (CESA'98)*, pages 857–861, Hammamet, Tunisia.
- Buss, M. (2000). *Control Methods for Hybrid Dynamical Systems – Models, Control Loops, Optimal Control, Computation Tools, and Mechatronic Applications – (in German)*. PhD thesis, Institute of Automatic Control Engineering, Technische Universität München.
- Buss, M., Glocker, M., Hardt, M., von Stryk, O., Bulirsch, R., and Schmidt, G. (2002). Nonlinear hybrid dynamical systems: Modeling, optimal control, and applications. In Engell, S., Frehse, G., and Schnieder, E., editors, *Modelling, Analysis, and Design of Hybrid Systems*, Lecture Notes in Control and Information Science. Springer. (This volume).
- Buss, M., Hashimoto, H., and Moore, J. (1996). Dextrous Hand Grasping Force Optimization. *IEEE Transactions on Robotics and Automation*, 12(3):406–418.
- Buss, M., Schlegl, T., and Schmidt, G. (1997). Development of Numerical Integration Methods for Hybrid (Discrete-Continuous) Dynamical Systems. In *Advanced Intelligent Mechatronic AIM97*, Tokyo, Japan.
- Buss, M. and Schmidt, G. (1996). Hybrid System Behavior Specification for Multiple Robotic Mechanisms. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems IROS*, pages 140–147, Osaka, Japan.
- Buss, M., von Stryk, O., Bulirsch, R., and Schmidt, G. (2000a). Towards hybrid optimal control. *at-Automatisierungstechnik*, 48:448–459.
- Buss, M., von Stryk, O., Bulirsch, R., and Schmidt, G. (2000b). Towards hybrid optimal control. *Automatisierungstechnik*, 9:448–459.
- Cellier, F., Elmqvist, H., and Otter, M. (1996). Modelling from physical principles. In Levine, W., editor, *The Control Handbook*, pages 99–107. CRC Press, Boca Raton, FL.
- Champagnat, R., Esteban, P., Pingaud, H., and Valette, R. (1996). Petri Net Based Modeling of Hybrid Systems. In *Proc. of ASI'96*, pages 53–60, Toulouse, France. Advanced Summer Institute.
- Champagnat, R., Esteban, P., Pingaud, H., and Valette, R. (1998). Modeling and Simulation of a Hybrid System Through PR/TR PN-DAE Model. In *Proc. of the 3rd Int. Conf. on Automation of Mixed Processes*, pages 131–137, Reims, France.
- Chase, C., Serrano, L., and Ramadge, P. J. (1993). Periodicity and chaos from switched flow systems: examples of discretely controlled continuous systems. *IEEE Trans. Automatic Control*.

- Cherif, M. and Gupta, K. K. (1997). Practical Motion Planning for Dextrous Re-Orientation of Polyhedra. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems IROS*, pages 291–297, Grenoble, Frankreich.
- Chouikha, M. (1999). *Entwurf diskret-kontinuierlicher Steuerungssysteme - Modellbildung, Analyse und Synthese mit hybriden Petri-Netzen*. PhD thesis, TU Braunschweig.
- Chouikha, M., Decknatel, G., Drath, R., Frey, G., Müller, C., Simon, C., Thieme, J., and Wolter, K. (2000). Petri net-based descriptions for discrete-continuous systems. *at - Automatisierungstechnik*, 48(9):415–425.
- Chouikha, M. and Krebs, V. G. (1998). Beschreibungsmittel und Methoden für kontinuierlich-diskrete Systeme. In Abel, D. and Lemmer, K., editors, *Theorie ereignisdiskreter Systeme*, München, Wien. Oldenbourg.
- Chouikha, M., Ober, B., and Schnieder, E. (2001). Automatisierter Steuerungsentwurf für diskrete und kontinuierlich-diskrete Systeme. *at - Automatisierungstechnik*, 49(6):280–289.
- Chouikha, M. and Schnieder, E. (1998a). Beschreibung kontinuierlich-diskreter Systeme mit hybriden Petrinetzen. In *GMA-Kongress '98 Mess- und Automatisierungstechnik*, pages 365–372, Ludwigsburg. Institut für Regelungs- und Automatisierungstechnik, TU Braunschweig, VDI-Verlag. VDI-Bericht 1397.
- Chouikha, M. and Schnieder, E. (1998b). Modelling of Continuous-discrete Systems with hybrid Petri Nets. In *IEEE: Computational Engineering in Systems Applications*, pages 606–612.
- Chouikha, M. and Schnieder, E. (1999). Model-based control synthesis of continuous-discrete systems. In *Proc. IEEE Int. Conf. Systems, Man and Cybernetics*, pages 452–456.
- Chow, A.-H. (1996). Parallel DEVS: A parallel, hierarchical, modular modeling formalism and its distributed simulator. *Transaction of the SCS International*, 13(2):55–67.
- Christen, E. (1997). The VHDL 1076.1 Language for Mixed-Signal Design. http://www.analogy.com/support/wp/vhdl_ern.htm.
- Chutinan, A. and Krogh, B. H. (1999a). Computing approximating automata for a class of linear hybrid systems. In *Hybrid Systems V: Proc. Int. Workshop, Notre Dame, USA*, Lecture Notes in Computer Science 1567, pages 16–37. Springer.
- Chutinan, A. and Krogh, B. H. (1999b). Verification of polyhedral-invariant hybrid automata using polygonal flow pipe approximation. In *2nd Int. Workshop on Hybrid Systems: Computation and Control*, volume 1569 of *LNCS*, pages 76–90. Springer.
- Ciardo, G., Nicol, D., and Trivedi, K. (1999). Discrete-event simulation of fluid stochastic petri nets. *IEEE Trans. Softw. Eng.*, 25(2):207–217.
- Clarke, E. M. and Emerson, E. A. (1982). Design and synthesis of synchronization skeletons for branching time temporal logic. In Kozen, D., editor, *Logics of Programs Workshop, IBM Watson Research Center, Yorktown Heights, New York, May 1981*, volume 131 of *Lecture Notes in Computer Science*, pages 52–71. Springer.

- Clarke, E. M., Grumberg, O., and Peled, D. A. (1999). *Model Checking*. MIT Press.
- Clarke, E. M. and Kurshan, R. P. (1996). Computer-aided verification. *IEEE Spectrum*, pages 61–67.
- Collins, D. (1995). *Designing Object-Oriented User Interfaces*. Benjamin/Cummings Publishing Company, Inc., Redwood City, CA.
- Console, L., de Kleer, J., and Hamscher, W., editors (1992). *Readings in Model-based Diagnosis*, San Mateo, CA. Morgan Kaufmann Publishers.
- Courcoubetis, C., Vardi, M. Y., Wolper, P., and Yannakakis, M. (1992). Memory-efficient algorithms for the verification of temporal properties. *Formal Methods in System Design*, 1(2/3):275–288.
- Cury, J. E. R., Krogh, B. A., and Niinomi, T. (1998). Synthesis of supervisory controllers for hybrid systems based on approximating automata. *IEEE Transactions on Automatic Control, Special issue on hybrid systems*, 43:564–568.
- Czogalla, O. and Hoyer, R. (1997). Simulation based design of control strategies for urban management and control. In *4th World Congress on Intelligent Transport Systems*, Berlin.
- Czogalla, O. and Hoyer, R. (1999). Model based approximation of traffic actuated signal control for mesoscopic traffic simulation. In *6th World Congress on Intelligent Transport Systems*, Toronto.
- Dang, T. and Maler, O. (1998). Reachability analysis via face lifting. In Henzinger, T. and Sastry, S., editors, *Hybrid Systems: Computation and Control, Proc. 1st Int. Workshop, HSCC'98, Berkeley, USA, March 1998*, Lecture Notes in Computer Science 1386, pages 96–109. Springer.
- David, R. and Alla, H. (1987). Continuous Petri Nets. In *8th European Workshop on Applications and Theory of Petri Nets*, pages 275–294, Spain.
- David, R. and Alla, H. (1992). *Petri nets and Grafcet - Tools for modelling discrete event systems*. Prentice Hall, New York, London.
- David, R. and Alla, H. (1994). Petri Nets for Modeling of Dynamic Systems - A Survey. *Automatica*, 30(2):175–202.
- David, R. and Alla, H. (1998). Continuous and hybrid Petri nets. *International Journal of Circuits and Systems*, 8(1):159–188.
- Davoren, J. M. and Nerode, A. (2000). Logics for hybrid systems. *Proceedings of the IEEE*, 88:985–1010.
- de Kleer, J. and Weld, D. S., editors (1990). *Readings in Qualitative Reasoning about Physical Systems*, San Mateo, CA. Morgan Kaufmann Publishers.
- de Roever, W.-P. (1998). The need for compositional proof systems: A survey. In de Roever, W.-P., Langmaack, H., and Pnueli, A., editors, *Compositionality: The Significant Difference, Proceedings of the International Symposium COMPOS '97, Malente, Germany, September 7–12, 1997*, volume 1536 of *Lecture Notes in Computer Science*, pages 1–22. Springer.
- de Roever, W.-P., de Boer, F., Hannemann, U., Hooman, J., Lakhnech, Y., Poel, M., and Zwiers, J. (2001). *Concurrency Verification: Introduction to Compositional and Noncompositional Methods*. Number 54 in Cambridge Tracts in Theoretical Computer Science. Cambridge University Press.
- De Schutter, B. (1999). Optimal control of a class of linear hybrid systems with saturation. In *Proc. 38th IEEE Conf. Decision and Control*, pages 3978–3983.

- Decknatel, G. and Schnieder, E. (1998). Hybrid petri nets as a new formalism for modelling railway systems. In *Computers in Railways VI*, pages 773–782. Computational Mechanics Publications/WIT Press.
- DEDS'98 (1998). Special issue on hybrid systems. *Discrete Event Dynamic Systems: Theory and Application*, 8:99–222.
- Denk, J. (1999). Online optimal control strategies for mechatronic systems under multiple contact configurations. Technical report, Institute of Automatic Control Engineering, Technische Universität München. Internal Report.
- Deperade, A., Pereira Remelhe, M., and Engell, S. (2001). Eine Modellierungs- und Simulationsumgebung für hybride technische Systeme mit ereignis-diskreten Steuerungen. In *3. VDI/VDE-GMA Aussprachetag, Rechnergestützter Entwurf von Regelungssystemen, Dresden*, volume 36 of *GMA-Berichte*, Düsseldorf. GMA-Aussprachetag FA-6.23, VDI/VDA-GMA.
- Design/CPN (2002). Design/CPN Version 4.0.1. <http://www.daimi.au.dk/design-CPN>. University of Aarhus, Department of Computer Science, CPN Group.
- Dijkstra, E. W. (1969a). On understanding programs (EWD 264). Published in an extended version as (Dijkstra, 1969b).
- Dijkstra, E. W. (1969b). Structured programming. In Buxton, J. and Randell, B., editors, *Software Engineering Techniques, Report on a conference sponsored by the NATO Science Committee*, pages 84–88. NATO Science Committee.
- Dill, D. (1990). Timing assumptions and verification of finite-state concurrent systems. In Sifakis, J., editor, *International Workshop on Automatic Verification Methods for Finite State Systems, Grenoble, France, June 12–14, 1989*, volume 407 of *Lecture Notes in Computer Science*, pages 197–212. Springer.
- Dimitriadis, V., Shah, N., and Pantelides, C. (1997). Modelling and safety verification of discrete/continuous processing systems. *AIChE Journal*, 43(4):1041–1059.
- Dimitriadis, V. D., Shah, N., and Pantelides, C. C. (1996a). A case study in hybrid process safety verification. *Computers and Chem. Eng.*, 20, Suppl.:S503–S508.
- Dimitriadis, V. D., Shah, N., and Pantelides, C. C. (1996b). Optimal design of hybrid controllers for hybrid process systems. In R. Alur, T.A. Henzinger, and E.D. Sontag, editors, *Lecture Notes in Computer Science 1066: Hybrid Systems III*, volume 1066 of *LNCS*, pages 224–257. Springer.
- Doğruel, M. and Özgüner, U. (1995). Modeling and stability issues in hybrid systems. In Antsaklis, P., Kohn, W., Nerode, A., and Sastry, S., editors, *Lecture Notes in Computer Science: Hybrid Systems II*, volume 999, pages 148–165. Springer.
- Doğruel, M., Özgüner, U., and Drakunov, S. (1996). Sliding-mode control in discrete-state and hybrid systems. *IEEE Transactions on Automatic Control*, 41:414–419.
- DoME (1999). DoME guide. <http://www.htc.honeywell.com/dome/>, Honeywell Technology Center, Honeywell. version 5.2.1.
- Drath, R. (1999). *Modellierung hybrider Systeme auf der Basis modifizierter Petri-Netze*. PhD thesis, TU-Ilmenau, Fachgebiet Automatisierungsanlagen und Prozeßleittechnik, Ilmenau. ISBN-Nr.: 3-932633-40-7.

- Drath, R., Engmann, U., and Schwuchow, S. (1999). Hybrid aspects of modelling manufacturing systems using modified petri nets. In *5th Workshop on Intelligent Manufacturing Systems*, Granado, Brasil.
- Drath, R. and Schwuchow, S. (1997). Modellierung diskret-kontinuierlicher Systeme mit Petri-Netzen. In Schnieder, E., editor, *Entwurf komplexer Automatisierungssysteme 5. Fachtagung*, pages 265–283, Braunschweig.
- Dymola (2002). Dymola version 4.2a. <http://www.dynasim.se/>.
- Elmqvist, H., Cellier, F. E., and Otter, M. (1993). Object-oriented modeling of hybrid systems. In *ESS'93, European Simulation Symposium*, Delft.
- Engell, S. (1997). Modellierung und Analyse hybrider dynamischer Systeme. *at-Automatisierungstechnik*, 45(4):152–162.
- Engell, S., editor (2000). *Special Issue on Discrete Event Models of Continuous Systems*, volume 6, no. 1 of *Mathematical and Computer Modelling of Dynamical Systems*.
- Engell, S., Hoffmann, I., and Sapronowa, L. (1997). Chaos in einfachen kontinuierlich-diskreten dynamischen Systemen. *at-Automatisierungstechnik*, 45(9):399–406.
- Engell, S., Kowalewski, S., and Zaytoon, J., editors (2000). *4th Int. Conf. on Automation of Mixed Processes: Hybrid Dynamic Systems (ADPM 2000)*, Dortmund, Germany. Shaker.
- Enste, U. (2001). *VDI Fortschritt-Berichte, Reihe 8, Nr. 884, Generische Entwurfsmuster in der Funktionsbauteiltechnik und deren Anwendung in der operativen Prozessführung*. VDI-Verlag.
- Enste, U. and Eppler, U. (1998). Standardisierte Prozessführungsbausteine - die Basis fuer Applikationsmodelle zur operativen Führung von verfahrenstechnischen Produktionsanlagen. In *VDI Bericht 1397*. VDI-Verlag.
- Enste, U. and Eppler, U. (2001). Technical application of hybrid modeling methods to specify function block systems. *Automatisierungstechnik - at*, 49(2):52–59.
- Enste, U. and Fedai, M. (1998). Flexible process control structures in multi-product and redundant-routing-plants. In *9th IFAC Symposium on Automation in Mining, Mineral and Metal Processing*, pages 211–214.
- Enste, U. and Kneissl, M. (2000). Modelling of software structures in process control systems - avoiding bugs by using graph grammars. In *IMACS Symposium on MATHEMATICAL MODELLING, ARGESIM Report No. 15: Proceedings Vol.1, Vienna*, pages 381–384.
- Enste, U. and Uecker, F. (2000). Use of supervision information in process control. *IEE Computing & Control Engineering Journal*, pages 234–241.
- Eppler, U. (1994). Operational control of process plants. In *Process Control Engineering*. VCH-Verlagsgesellschaft, Weinheim.
- Ernst, T., Jähnichen, S., and Klose, M. (1997). Object-oriented physical systems modeling, Modelica, and the SmileM simulation environment. In Sydow, A., editor, *Proceedings of the 15th IMACS World Congress on Scientific Computation, Modelling and Applied Mathematics*, volume 6, pages 653–658.
- Ernst, T., Klein-Robbenhaar, C., Nordwig, A., and Schrag, T. (2000). Modellierung und Simulation hybrider Systeme mit Smile. *Informatik Forschung und Entwicklung*, 5.

- Evans, R. and Savkin, A., editors (1999). *Systems and Control Letters, Special issue on Hybrid Control Systems*, volume 38(3).
- Fábián, G., van Beek, D. A., and Rooda, J. E. (1998). Integration of the discrete and the continuous behaviour in the hybrid chi simulator. In *1998 European Simulation Multiconference, Manchester*, pages 207–257.
- Fahrland, D. (1970). Combined discrete event continuous systems simulation. *Simulation*, 14(2):71–72.
- Fellendorf, M. (1994). VISSIM: A microscopic Simulation Tool to evaluate Actuated Signal Control including Bus priority. In *64th ITE Annual Meeting*, Dallas.
- Fieldbus DDL (1996). Device description language specification. Technical report, Fieldbus Foundation, Austin Texas.
- Floyd, R. W. (1967). Assigning meanings to programs. In Schwartz, J., editor, *Proceedings AMS Symposium Applied Mathematics*, volume 19, pages 19–31, Providence, RI. American Mathematical Society.
- Föllinger, O. (1994). *Regelungstechnik. Einführung in die Methoden und ihre Anwendung*. Hüthig.
- Forbus, K. D. (1990). Qualitative reasoning. Draft chapter.
- Förstner, D. (2001). *Qualitative Modellierung für die Prozeßdiagnose und deren Anwendung auf Dieseleinspritzpumpen*. PhD thesis, TU Hamburg-Harburg.
- Frank, P. M. (1998). Komplexe Systeme - Nichtlineare Rückkopplungssysteme jenseits der Stabilität. *at - Automatisierungstechnik*, 46(4):167–179.
- Frank, R. (2001). Entwicklung einer Internetanbindung für den Modellprozess Drei-Tank-System. Diplomarbeit, Institut für Automatisierungs- und Softwaretechnik (IAS), Universität Stuttgart.
- Franke, D., Moor, T., and Raisch, J. (2000). Discrete supervisory control of switched linear systems. *at-Automatisierungstechnik*, 48:9:461–467.
- Frehse, G., Stursberg, O., Engell, S., Huuck, R., and Lukoschus, B. (2002). Modular analysis of discrete controllers for distributed hybrid systems. In *b'02: The XV. IFAC World Congress, Barcelona, Spain, July 21–26, 2002*. To appear.
- Frehse, G. F., Stursberg, O., Engell, S., Huuck, R., and Lukoschus, B. (2001). Verification of hybrid controlled processing systems based on decomposition and deduction. In *ISIC 2001: 16th IEEE International Symposium on Intelligent Control, Mexico City, Mexico, September 5–7, 2001*, pages 150–155. IEEE Control Systems Society, IEEE Press.
- Friesen, V. (1995). An exercise in hybrid system specification using an extension of Z. In Bouajjani, A. and Maler, O., editors, *Second European Workshop on Real-Time and Hybrid Systems*, pages 311–316.
- Friesen, V. (1997). *Objektorientierte Spezifikation hybrider Systeme*. PhD thesis, Technical University of Berlin.
- Friesen, V. (1998). A logic for the specification of continuous systems. LNCS 1386, Berlin, Germany. Springer.
- Friesen, V., Nordwig, A., and Weber, M. (1998a). Object-oriented specification of hybrid systems using UML^h and ZimOO. In *Proc. 11th Int. Conf. on the Z Formal Method (ZUM)*, LNCS 1493. Springer.

- Friesen, V., Nordwig, A., and Weber, M. (1998b). Toward an object-oriented design methodology for hybrid systems. *Proceedings of the Colloquium on Object Technology and System Re-Engineering*, Oxford.
- Fröhlich, P. (1996). *Überwachung verfahrenstechnischer Prozesse unter Verwendung eines qualitativen Modellierungsverfahrens*. PhD thesis, Institut für Automatisierungs- und Softwaretechnik (IAS), Universität Stuttgart.
- Gamma, E., Helm, R., Johnson, R., and Vlissides, J. (1995). *Design Patterns, Elements of Reusable Object-Oriented Software*. Addison-Wesley.
- Gao, Z. and Antsaklis, P. J. (1991). Stability of the pseudo-inverse method for reconfigurable control systems. *International Journal of Control*, 53:717–729.
- Gazis, D. C. et al. (1959). Car following theory of steady-state traffic flow. *Operns. Res.*, 7:499–505.
- Geisler, R., Klar, M., and Pons, C. (1998). Dimensions and dichotomy in metamodeling. Technical Report 98-2, Technical University of Berlin.
- Genrich, H. J. (1978). Ein Kalkül des Planens und Handelns. In *Ansätze zur Organisationstheorie rechnergestützter Informationssysteme*, GMD Bericht 111, pages 77–92. Oldenbourg.
- Genrich, H. J. (1987). Predicate/transition nets. *Advances in Petri nets 1986, part I. Lecture Notes in Computer Science*, 254:207–247.
- Genrich, H. J. and Lautenbach, K. (1981). System modelling with high-level petri nets. *Theoretical Computer Science*, 13.
- Ghezzi, C., Mandrioli, D., Morasca, S., and Pezzè, M. (1991). A unified high-level petri net formalism for time-critical systems. *IEEE Transactions On Software Engineering*, 17(2):160–172.
- Gill, P., Murray, W., and Saunders, M. (1997). *User's guide for SNOPT 5.3: a fortran package for large-scale nonlinear programming*. Department of Mathematics, Univ. of California San Diego.
- Gilles, E. D., Holl, P., and Marquardt, W. (1986). Dynamische Simulation komplexer chemischer Prozesse. *Chem.-Ing.-Tech*, 58(4):268–278.
- Giua, A. and Piccaluga, A. (2002). Bibliography on hybrid petri nets. <http://bode-diee.unica.it/~hpn/>.
- Giua, A. and Usai, E. (1996). High-level hybrid petri nets: a definition. In *35th Conference on Decision and Control*, pages 148–150, Kobe, Japan.
- Giua, A. and Usai, E. (1998). Modeling hybrid systems by high-level petri nets. In *ADPM'98*, pages 316–323.
- Glocker, C. (1995). *Dynamik von Starrkörpersystemen mit Reibung und Stößen*. PhD thesis, TU München, München.
- Glover, F. (1975). Improved linear integer programming formulations of nonlinear integer problems. *Managem. Science*, 22(4):455–460.
- Göhner, P. and Lauber, R. (1999). *Prozessautomatisierung 2*, volume 2. Springer, Berlin Heidelberg, 1 edition.
- Gokbayrak, K. and Cassandras, C. G. (2000). Hybrid controllers for hierarchically decomposed systems. In *Proc. 3rd Int. Workshop of Hybrid Systems: Computations and Control*, volume 1790 of *LNCS*, pages 117–129. Springer.

- Goldstein, H. H. and von Neumann, J. (1947). Planning and coding problems of an electronic computing instrument. In Taub, A., editor, *J. von Neumann—Collected Works*, pages 80–151. McMillan, New York.
- gPROMS (2002). Homepage: <http://www.psenterprise.com/>.
- Greenstreet, M. and Mitchell, I. (1999). Reachability analysis using polygonal projections. In Vaandrager, F. W. and van Schuppen, J. H., editors, *Hybrid Systems: Computation and Control, Proc. 2nd Int. Workshop, HSCC'99, Berg en Dal, The Netherlands, March 1999*, Lecture Notes in Computer Science 1569, pages 103–116. Springer.
- Gribaudo, M., Sereno, M., and Bobbio, A. (1999). Fluid stochastic petri nets: An extended formalism to include non-markovian models. In *Proc. Eighth International Workshop on Petri Nets and Performance Models - PNPM'99*, pages 74–81, Zaragoza, Spain.
- Griepentrog, E. and März, R. (1986). *Differential-Algebraic Equations and Their Numerical Treatment*. BSB Teubner, Leipzig. ISBN 3-322-00343-4.
- Grossman, R. L., Nerode, A., Ravn, A. P., and Rischel, H., editors (1993). *Hybrid Systems*, volume 736 of *Lecture Notes in Computer Science*. Springer.
- Grosu, R., Krüger, I., and Stauner, T. (2000). Hybrid Sequence Charts. In *Proc. of ISORC 2000*. IEEE.
- Grosu, R., Stauner, T., and Broy, M. (1998). A modular visual model for hybrid systems. In *Proc. of FTRTFT'98*, LNCS 1486. Springer.
- Group, I. . W. (1999). IEEE standard 1076.1-1999. <http://www.vhdl.org>.
- Haidacher, S., Schlegl, T., and Buss, M. (1999). Grasp Evaluation Based on Unilateral Force Closure. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems IROS*, pages 424–429, Kyongju, Korea.
- Hamscher, W., deKleer, J., and Console, L., editors (1992). *Readings in Model-Based Diagnosis*. Morgan Kaufman.
- Hanisch, H.-M. (1992). *Petri-Netze in der Verfahrenstechnik*. Oldenbourg, München, Wien.
- Hanisch, H.-M., Lautenbach, K., Simon, C., and Thieme, J. (1998a). Timestamp nets in technical applications. In *IEEE International Workshop on Discrete Event Systems*, San Diego, CA, USA.
- Hanisch, H.-M., Lautenbach, K., Simon, C., and Thieme, J. (1998b). Timestamp petri nets in technical applications. In Giua, A., Smedinga, R., and Spathopoulos, M. P., editors, *IEE International Workshop on Discrete Event Systems*, IEE Control, pages 321–326, Cagliari, Sardinia, Italy.
- Hanisch, H.-M., Lautenbach, K., Simon, C., and Thieme, J. (1998c). Zeitstempelnetze in technischen Anwendungen. Fachberichte Informatik 2–98, Universität Koblenz-Landau, Institut für Informatik, Rheinau 1, D-56075 Koblenz.
- Hardt, M., Helton, J., and Kreutz-Delgado, K. (2000). Numerical solution of nonlinear \mathcal{H}_2 and \mathcal{H}_∞ control problems with application to jet engine compressors. *IEEE Transactions on Control Systems Technology*, 8(1):98–111.
- Hardt, M. and von Stryk, O. (2000). Towards optimal hybrid control solutions for gait patterns of a quadruped. In *CLAWAR 2000 – 3rd International Conference on Climbing and Walking Robots, Madrid, 2–4 October, Professional Engineering Publishing, UK*, pages 385–392.

- Harel, D. (1987). Statecharts: A Visual Formalism for Complex Systems. *Science of Computer Programming*, 8:231–274.
- Harel, D. and Gery, E. (1996). Executable object modeling with Statecharts. In *Proceedings of the 18th International Conference of Software Engineering*, IEEE Press.
- Harel, D., Pnueli, A., Schmidt, J., and Sherman, R. (1987). On the formal semantics of statecharts. In *2nd IEEE Symp. on Logic in Computer Science*, pages 54–64. IEEE Press.
- He, K. X. and Lemmon, M. D. (1998). Lyapunov Stability of Continuous-Valued Systems Under the Supervision of Discrete-Event Transition Systems. In Henzinger, T. A. and Sastry, S., editors, *Hybrid Systems: Computation and Control*, LNCS 1386, pages 175–189, Berlin, Germany. Springer.
- Hedlund, S. and Rantzer, A. (1999). Optimal control of hybrid systems. In *Proceedings of the 38th IEEE Conference on Decision and Control (Phoenix, AZ)*, pages 3972–3977.
- Heinkel, U. (2000). *The VHDL reference*. Wiley, Chichester.
- Henzinger, A., Kopke, P., Puri, A., and Varaiya, P. (1995). What’s decidable about hybrid automata. In *Proceedings of the 27th Annual ACM Symposium on Theory of Computing (STOC1995)*, pages 373–382.
- Henzinger, T., Ho, P., and Wong-Toi, H. (1997). Hytech: A model checker for hybrid systems. *Software Tools for Technology Transfer*, 1(1,2):110–122.
- Henzinger, T., Ho, P., and Wong-Toi, H. (1998a). Algorithmic analysis of nonlinear hybrid systems. *IEEE Transactions on Automatic Control*, 43(4):540–554.
- Henzinger, T., Kopke, P., Puri, A., and P. Varaiya (1998b). What’s decidable about hybrid automata. *J. Comp. Syst. Science*, 57:94–124.
- Henzinger, T., Qadeer, S., Rajamani, S., and Tasiran, S. (1998c). You assume, we guarantee: Methodology and case studies. In *Proc. 10th Int. Conf. on Computer-Aided Verification*, volume 1427 of *Lecture Notes in Computer Science*, pages 440–451. Springer.
- Henzinger, T. A. (1996). The theory of hybrid automata. In *Proc. of 11th Annual IEEE Symposium on Logic in Computer Science (LICS’96)*, pages 278–292. IEEE Computer Society Press.
- Henzinger, T. A., Minea, M., and Prabhu, V. (2001). Assume-guarantee reasoning for hierarchical hybrid systems. In *HSCC ’01: 4th International Workshop on Hybrid Systems: Computation and Control*, volume 2034 of *Lecture Notes in Computer Science*, pages 275–290. Springer.
- Henzinger, T. A. and Sastry, S., editors (1998). *Hybrid Systems – Computation and Control (HSCC’98)*, volume 1386 of *Lecture Notes in Computer Science*. Springer.
- HLA TMD Document (1996). HLA time management design document, version 1.0.
- Hoare, C. (1969). An axiomatic basis for computer programming. *Communications of the ACM*, 12(10):576–580, 583.
- Hoare, C. (1985). *Communicating Sequential Processes*. Prentice-Hall International, Englewood Cliffs.

- Holzmann, G. J. (1997). The model checker SPIN. *IEEE Transactions on Software Engineering*, 23(5):279–295.
- Horton, G., Kulkarni, V. G., Nicol, D. M., and Trivedi, K. S. (1998). Fluid stochastic petri nets: Theory, applications and solution. *European Journal of Operations Research*, 105(1):184–201.
- Huber, F., Schätz, B., and Einert, G. (1997). Consistent graphical specification of distributed systems. In *FME '97: 4th International Symposium of Formal Methods Europe, LNCS 1313*, pages 122 – 141.
- Hubert, P., Jensen, K., and Shapiro, R. (1991). Hierarchies in coloured petri nets. *Lecture Notes in Computer Science*, 483.
- Huuck, R., Engell, S., Kowalewski, S., Lakhnech, Y., Preußig, J., and Urbina, L. (1997). Comparing timed c/e systems with timed automata. In *International Workshop on Hybrid and Real-Time Systems (Hart '97)*, LNCS 1201, pages 81–86, Grenoble. Springer.
- IEC 1131 (1993). International standard IEC 1131 programmable controllers, part 3, programming languages.
- IEC 61131-3 (1992). Programming language for programmable controllers. Technical report, Committee IEC 61131-3.
- IEC SC 65C WG7 (1999). Function blocks for process control. Technical report, Committee IEC 61804-1.
- IEC TC65 WG6 (1999). Function blocks for industrial-process measurement and control systems. Technical report, Committee IEC 61499-1.
- Ioannou, P. (1996). *Robust Adaptive Control*. Prentice-Hall Upper Saddle River NJ.
- Isermann, R. (1996a). Modellgestützte Überwachung und Fehlerdiagnose Technischer Systeme (Teil 1). *atp*, 38(5):9–20.
- Isermann, R. (1996b). Modellgestützte Überwachung und Fehlerdiagnose Technischer Systeme (Teil 2). *atp*, 38(6):48–57.
- ITU (1999). ITU-T Recommendation Z.120: Message Sequence Charts (MSC).
- Jähnichen, S. and Klein-Robbenhaar, C. (2000). Generic modeling and simulation of hybrid systems with adaptive modeling depth. Technical report, Technical University of Berlin. (in German).
- Jensen, H. E. (1999). *Abstraction-Based Verification of Distributed Systems*. PhD thesis, Aalborg University.
- Jensen, H. E., Larsen, K. G., and Skou, A. (2000). Scaling up Uppaal – automatic verification of real-time systems using compositionality and abstraction. In Joseph, M., editor, *FTRTFT 2000: 6th International Symposium on Formal Techniques in Real-Time and Fault-Tolerant Systems, September 20–22, 2000, Pune, India*, volume 1926 of *Lecture Notes in Computer Science*, pages 19–30. Springer.
- Jensen, K. (1992). *Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use*, volume 1. Springer.
- Jensen, K. (1997). *Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use*, volume 2. Springer.
- Jensen, K. and Rozenberg, G., editors (1991). *High-level Petri Nets: theory and application*. Springer.

- Jhala, R. and McMillan, K. L. (2001). Microarchitecture verification by compositional model checking. In Berry, G., Comon, H., and Finkel, A., editors, *CAV 2001: 13th International Conference on Computer Aided Verification, Paris, France, July 18–22, 2001*, volume 2102 of *Lecture Notes in Computer Science*, pages 396–410. Springer.
- Jirstrand, M. (1998). *Constructive Methods for Inequality Constraints in Control*. PhD thesis, Department of Electrical Engineering, Linköping University, Linköping, Sweden.
- Johansson, M. and Rantzer, A. (1998). Computation of Piecewise Quadratic Lyapunov Functions for Hybrid Systems. *IEEE Trans. Aut. Control*, 43(4):555–559.
- John, S. (2001). Transition selection algorithms for Statecharts. *Proceedings of the GI/OCG annual congress*, 1:pp. 622–627.
- Jones, C. B. (1981). *Development Methods for Computer Programs including a Notion of Interference*. PhD thesis, Oxford University Computing Laboratory. Printed as: Programming Research Group, Technical Monograph 25.
- Jones, C. B. (1983). Tentative steps toward a development method for interfering programs. *ACM Transactions on Programming Languages and Systems*, 5(4):596–619.
- Joos, H.-D. (1999). A methodology for multi-objective design assessment and flight control synthesis tuning. *Aerospace Science and Technology*, 3(3):161–176.
- Kaiser, R. and Beaumariage, T. (1997). Conceptual design of an artificial intelligence architecture for decision making in manufacturing simulation. In Wallace, J. and Beaumariage, T., editors, *Object-Oriented Simulation Conf. OOS'97*, pages 11–15. SCS International, San Diego.
- Kienle, A. (2000). Low-order models for ideal multicomponent distillation processes using nonlinear wave propagation theory. *Chemical Engineering Science*, 55:1817–1828.
- Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, 220(4598):671–680.
- Klar, M. and Mann, S. (1998). A Metamodel for Object-Oriented Statecharts. *The Second Workshop on Rigorous Object Oriented Methods, ROOM 2*.
- Klein, E., Itigin, A., Raisch, J., and Kienle, A. (2000). Automatic generation of switching start-up schemes for chemical processes. *Proc. ESCAPE10 – 10th European Symposium on Computer Aided Process Engineering*, pages 619–624.
- Klein, E., Kienle, A., and Raisch, J. (1998). Synthesizing a supervisory control scheme for the start-up procedure of a distillation column - an approach based on approximating continuous dynamics by des models. *Proc. LSS'98 - 8th IFAC Colloquium on Large Scale Systems*, pages 716–721.
- Klein, E., Kienle, A., Raisch, J., and Wehlan, H. (1999). Synthese einer Anfahrregelung für eine Destillationskolonne auf der Grundlage einer ereignisdiskreten Approximation der kontinuierlichen Dynamik. *6. Fachtagung Entwicklung und Betrieb komplexer Automatisierungssysteme (EKA99)*, pages 447–464.

- Klein, E. and Raisch, J. (1998). Safety enforcement in process control systems - a batch evaporator example. In *Proc. WODES'98 - International Workshop on Discrete Event Systems, Cagliari, Italy*, pages 327–333. IEE.
- Kleinmann, K. P. (1996). *Lernende Regelung eines mehrfingrigen Robotergräfers*. PhD thesis, TU Darmstadt, Darmstadt.
- Kloas, M., Friesen, V., and Simons, M. (1995). Smile — A simulation environment for energy systems. In Sydow, A., editor, *Proceedings of the 5th International IMACS-Symposium on Systems Analysis and Simulation (SAS'95)*, pages 503–506. Gordon and Breach Publishers.
- Komarow, W. B. and Skotschinski, A. A. (1956). *Grubenbewetterung*. VEB Verlag Technik Berlin.
- Kondak, K. and Hommel, G. (2001). Computation of time optimal movements for autonomous parking of non-holonomic mobile platforms. In *Proceedings of the IEEE International Conference on Robotics and Automation (Seoul, Korea)*, pages 2698–2703.
- König, R. and Quäck, L. (1988). *Petri-Netze in der Steuerungs- und Digitaltechnik*. Oldenbourg, München, Wien.
- Koutsoukos, X., Antsaklis, P. J., Stiver, J. A., and Lemmon, M. D. (2000). Supervisory control of hybrid systems. *Proceedings of the IEEE*, 88:1026–1049.
- Kowalewski, S. (1996). *Modulare diskrete Modellierung verfahrenstechnischer Anlagen zum systematischen Steuerungsentwurf*. PhD thesis, Fachbereich Chemietechnik, Dortmund.
- Kowalewski, S. (2002). Introduction to the analysis and verification of hybrid systems. In *this volume*.
- Kowalewski, S., Engell, S., Preussig, J., and Stursberg, O. (1999). Verification of logic controllers for continuous plants using timed condition/event system models. *Automatica*, 35(3):505–518.
- Kowalewski, S., Herrmann, P., Engell, S., Huuck, R., Krumm, H., Lakhnech, Y., and Lukoschus, B. (2001a). Approaches to the formal verification of hybrid systems. *at-Automatisierungstechnik*, 49(2):66–74.
- Kowalewski, S. and Preußig, J. (1996). Timed condition/event systems: A framework for modular models of chemical plants and verification of their real-time discrete control. In Margaria, T. and Steffen, B., editors, *Tools and Algorithms for the Construction and Analysis of Systems, Proc. 2nd International Workshop TACAS'96*, Lecture Notes in Computer Science 1055, pages 225–240, Passau. Springer.
- Kowalewski, S., Stursberg, O., and Bauer, N. (2001b). An experimental batch plant as a test case for the verification of hybrid systems. *European Journal of Control*, 7.
- Kowalewski, S., Stursberg, O., and Treseler, H. (1998). Diskrete Modellierung verfahrenstechnischer Prozesse zur Steuerungsverifikation. *at - Automatisierungstechnik*, 4:180–187.
- Kramer, D. (1997). *JDK 1.1.1 Documentation*. Sun Microsystems, Inc.
- Krebs, V. G. and Schnieder, E., editors (2000). *Hybrid Systems I: Modeling and Control*, volume 48.

- Kripke, S. A. (1963). Semantical considerations on modal logic. *Acta Philosophica Fennica*, 16:83–94.
- Krogh, B. (1993). Condition/event signal interfaces for block diagram modeling and analysis of hybrid systems. In *8th Int. Symp. on Intelligent Control Systems*, pages 180–185.
- Kuipers, B. (1986). Qualitative simulation. *Artificial Intelligence*, 29:289–338.
- Kuipers, B. (1994). *Qualitative Reasoning*. MIT Press.
- Kupferman, O., Vardi, M. Y., and Wolper, P. (2000). An automata-theoretic approach to branching-time model checking. *Journal of the ACM*, 47(2):312–360.
- Kurz, H. (1990). Realisierung gehobener Methoden der Regelungstechnik auf Prozessleitsystemen - Ein Diskussionsbeitrag. *Automatisierungstechnische Praxis - atp*, 32(10):489–494.
- Labinaz, G., Bayoumi, M. M., and Rudie, K. (1996). Modeling and Control of Hybrid Systems: A Survey. In *Proc. IFAC 13th Triennial World Congress*, pages 293–304, San Francisco, USA. IFAC.
- Lafferriere, G., Pappas, G., and Yovine, S. (1999). A new class of decidable hybrid systems. In Vaandrager, F. W. and van Schuppen, J. H., editors, *Hybrid Systems: Computation and Control, Proc. 2nd Int. Workshop, HSCC'99, Berg en Dal, The Netherlands, March 1999*, volume 1569 of *Lecture Notes in Computer Science*, pages 137–151. Springer.
- Lafferriere, G., Pappas, G., and Sastry, S. (2000). O-minimal hybrid systems. *Mathematics of Control, Signals, and Systems*, 13(3):1–21.
- Larsen, K. G., Pettersson, P., and Yi, W. (1997). UPPAAL in a nutshell. *International Journal on Software Tools for Technology Transfer*, 1(1–2):134–152.
- Laudwein, A. (1999). Konzeption und Entwicklung einer Steuerungs- und Regelungssoftware für den Modellprozess “Drei-Tank-System”. Diplomarbeit, Institut für Automatisierungs- und Softwaretechnik (IAS), Universität Stuttgart.
- Laufenberg, X. (1997). *Ein modellbasiertes qualitatives Verfahren für die Gefahrenanalyse*. Dissertation, Institut für Automatisierungs- und Softwaretechnik (IAS), Universität Stuttgart.
- Lautenbach, K. and Simon, C. (1999). Erweiterte Zeitstempelnetze. Fachberichte Informatik 03–99, Universität Koblenz-Landau, Institut für Informatik, Rheinau 1, D-56075 Koblenz.
- Lautenbach, K. and Simon, C. (2000). Verification in a logic of actions. In *7. Workshop Algorithmen und Werkzeuge für Petrinetze*, Koblenz.
- Lautenbach, K. and Simon, C. (2001). Modellierung der Dynamik einer Batch-Anlage. In Schnieder, E., editor, *Engineering komplexer Automatisierungssysteme, EKA 2001*, Braunschweig.
- Le Bail, J., Alla, H., and David, R. (1991). Hybrid petri nets. In *European Control Conference*, pages 1472–1477.
- Lee, J.-D. e. a. (2000). Analysis of moving and fixed autoblock systems for korean high speed railway. In *Computers in Railways VII*, pages 843–851. WIT Press, Bologna.

- Lee, S. and Grossmann, I. (2000). New algorithms for nonlinear generalized disjunctive programming. *Comp. and Chemical. Eng.*, 4:2125–2141.
- Lemmon, M., He, K., and Markovsky, I. (1999). Supervisory hybrid systems. *IEEE Control Systems Magazine*, 19:42–55.
- Leue, S., Mehrmann, L., and Rezai, M. (1998). Synthesizing ROOM Models from MSC Specifications. Technical Report TR-98-06, University of Waterloo.
- Levin, G. M. and Gries, D. (1981). A proof technique for communicating sequential processes. *Acta Informatica*, 15(3):281–302.
- Li, Z., Soh, C., and Xu, X. (2000). Lyapunov stability of a class of hybrid dynamic systems. *Automatica*, 36:297–302.
- Liberzon, D. and Morse, A. S. (1999). Basic problems in stability and design of switched systems. *IEEE Control Systems Magazine*, 19.
- Lichtenberg, G. and Kamau, S. (2001). A classification of the input-output behaviour of hybrid systems. In *European Control Conference*.
- Lichtenberg, G., Lunze, J., and Raisch, J. (1999a). Two approaches to modeling the qualitative behaviour of dynamic systems. *at-Automatisierungstechnik*, 47:187–198.
- Lichtenberg, G., Lunze, J., and Raisch, J. (1999b). Zwei Wege zur Modellierung des qualitativen Verhaltens dynamischer Systeme. *at - Automatisierungstechnik*, 47(5):187–198.
- Lichtenberg, G., Lunze, J., Scheuring, R., and Schröder, J. (1999c). Prozessdiagnose mittels qualitativer Modelle am Beispiel eines Wasserstoffverdichters. *at - Automatisierungstechnik*, 47(3):101–109.
- Lichtenstein, O. and Pnueli, A. (1985). Checking that finite state concurrent programs satisfy their linear specifications. In *Twelfth ACM Symposium on the Principles of Programming Languages*, pages 97–105.
- Liggesmeyer, P. and Mäkel, P. (2000). Automtisierung erweiterter Fehlerbaumanalysen für komplexe technische Systeme. *at - Automatisierungstechnik*, 48(2):67–76.
- Lighthill, M. J. and Whitham, G. B. (1955). On kinematic waves. ii. a theory of traffic flow on long crowded roads. In *Roy. Society*, volume 229 A, pages 317–345, London.
- Lincoln, B. and Rantzer, A. (2001). Optimizing linear system switching. In *Proc. 40th IEEE Conf. Decision and Control*, pages 2063–2068.
- Lorch, O., Denk, J., Seara, J., Buss, M., Freyberger, F., and Schmidt, G. (2000). Vigwam — an emulation environment for a vision guided virtual walking machine. In *Proceedings of the First IEEE-RAS International Conference on Humanoid Robots HUMANOIDS 2000 (Cambridge, MA, USA)*.
- Lötzbeier, H. and Pretschner, A. (2000). AutoFocus on Constraint Logic Programming. In *Proc. (Constraint) Logic Programming and Software Engineering*.
- Lunze, J. (1994). Qualitative modelling of linear dynamical systems with quantized state measurements. *Automatica*, 30:417–431.
- Lunze, J. (1995). Stabilisation of nonlinear systems by qualitative feedback controllers. *Intern. J. Control*, 62:109–128.
- Lunze, J. (1998a). On the Markov property of quantised state measurement sequences. *Automatica*, 34:1439–1444.

- Lunze, J. (1998b). Qualitative Modellierung dynamischer Systeme durch stochastische Automaten. *at - Automatisierungstechnik*, 46(6):271–283.
- Lunze, J. (1999). A timed discrete-event abstraction of continuous-variable systems. *Intern. J. Control*, 72:1147–1164.
- Lunze, J. (2000). Process supervision by means of qualitative models. *Annual Reviews in Control*, 24:41–54.
- Lunze, J. (2001). Control reconfiguration. In *Encyclopedia of Live Support Systems*. EOLSS Publishers. submitted.
- Lunze, J. (2002). *Regelungstechnik, Band 2*. Springer.
- Lunze, J., Heiming, B., and et. al., M. S. (2000). Three-tank control reconfiguration. In Aström, K., editor, *Control of Complex Systems*. Springer.
- Lunze, J. and Nixdorf, B. (2002). Representation of hybrid systems by means of stochastic automata. *Mathematical Modelling of Systems*, 7:383–422.
- Lunze, J. and Nixdorf, B. (2003). Discrete reachability of hybrid systems. *Intern. J. Control*, submitted.
- Lunze, J., Nixdorf, B., and Richter, H. (1997). Hybrid modelling of continuous-variable systems with application to supervisory control. In *Proceedings of the European Control Conference 1997*.
- Lunze, J. and Raisch, J. (2002). Discrete models for hybrid systems. In Engell, S., Frehse, G., and Schnieder, E., editors, *Modelling, Analysis, and Design of Hybrid Systems*, Lecture Notes in Control and Information Science. Springer. (This volume).
- Lunze, J. and Schiller, F. (1997). Qualitative Prozessdiagnose auf wahrscheinlichkeitstheoretischer Grundlage. *at - Automatisierungstechnik*, 45(8):351–359.
- Lunze, J. and Schröder, J. (1999). Process diagnosis based on a discrete-event description. *at - Automatisierungstechnik*, 47:358–365.
- Lunze, J. and Steffen, T. (2000). Reconfigurable control of a quantised system. In *Proceeding of SAFEPROCESS 2000: 4th Symposium on Fault Detection*, pages 822–827. IFAC.
- Lunze, J. and Steffen, T. (2002). Hybrid reconfigurable control. In *this volume*.
- Lüth, T. (1998). *Technical Multiagent Systems*. Hanser Publisher. (in German).
- Lygeros, J., Tomlin, C., and Sastry, S. (1999). Controllers for reachability specifications for hybrid systems. *Automatica*, 35:349–370.
- Lynch, N. and Krogh, B. H., editors (2000). *Hybrid Systems – Computation and Control (HSCC 2000)*, volume 1790 of *Lecture Notes in Computer Science*. Springer.
- Lynch, N., Segala, R., Vaandrager, F., and Weinberg, H. B. (1996). Hybrid I/O automata. In Alur, R., Henzinger, T. A., and Sontag, E. D., editors, *Hybrid Systems III*, LNCS 1066, pages 496–510. Springer.
- Maciejowski, J. (2002). *Predictive control with constraints*. Prentice Hall.
- Mai, G. and Schröder, M. (1999). Simulation of a Flight Control Systems' Redundancy Management System using Statemate. 7. User group meeting STATE-MATE.
- Maler, O., editor (1997). *Hybrid and Real-Time Systems (HART'97)*, volume 1201 of *Lecture Notes in Computer Science*. Springer.

- Maler, O., editor (2001). *Special Issue on Verification of Hybrid Systems*, volume 7, issue 4 of *European Journal of Control*.
- Manz, S. (1999). Qualitative Modeling of a Three-Tank-System. In *Interkama-ISA Tech Conference*, Düsseldorf.
- Manz, S. (2000). On-line monitoring and diagnosis based on hybrid component models. In *13th International Conference on Software & Systems Engineering and Applications ICSSEA 2000*, Paris.
- Manz, S. (2001a). Fuzzy based qualitative models in combination with dynamical models for online monitoring of technical systems. In *International Conference on Computational Intelligence for Modelling, Control and Aut. CIMCA 2001*, Las Vegas.
- Manz, S. (2001b). Online fault detection and diagnosis of complex systems based on hybrid component models. In *14th International Congress on Condition Monitoring and Diagnostics Engineering Managem. COMADEM2001*, Manchester.
- Mareczek, J., Buss, M., and Schmidt, G. (1998). Robust Global Stabilization of the Underactuated 2-DOF Manipulator R2D1. In *Proceedings of the IEEE International Conference on Robotics and Automation (Leuven, Belgium)*, pages 2640–2645.
- Mareczek, J., Buss, M., and Schmidt, G. (1999). Robust Control of a Non-Holonomic Underactuated SCARA Robot. In Tzafestas, S. and Schmidt, G., editors, *Lecture Notes in Control and Information Sciences: Progress in System and Robot Analysis and Control Design*, volume 243, pages 381–396. Springer.
- Marsan, M. A., Balbo, G., Chiola, G., Donatelli, S., and Francheschinis, G. (1995). *Modelling with Generalized Stochastic Petri Nets*. John Wiley & Sons.
- Martin, B. and Bobrow, J. (1997). Minimum effort motions for open chain manipulators with task-dependent end-effector constraints. In *Proceedings of the IEEE International Conference on Robotics and Automation (Albuquerque, New Mexico)*, pages 2044–2049.
- Matlab (2002). Homepage: <http://www.mathworks.com>.
- Matsuno, H. and Doi, A. (2000). Hybrid Petri Net Representation of Gene Regulatory Network. In *Pacific Symposium on BioComputing 2000*, pages 341–352, Hawaii.
- Matsuno, H., Doi, A., Drath, R., and Miyano, S. (2000). Genomic object net: Object representation of biological systems. *Genome Informatics*, 11.
- Matsuno, H., Doi, A., Drath, R., and Miyano, S. (2001). Genomic object net: Hybrid petri net for describing biological systems. In *Fifth Annual International Conference on Computational Molecular Biology*, Montreal, Canada.
- Matsuno, H. and Miyano, S. (2000). A platform for virtual cells; simulation of gene regulatory control by hybrid object net. *bit*, 32:22–31. (in Japanese).
- McMillan, K. L. (1992). *Symbolic Model Checking: An Approach to the State Explosion Problem*. PhD thesis, Carnegie Mellon University. CMU Technical Report CMU-CS-92-131.
- McMillan, K. L. (1995). A Technique of a State Space Search Based on Unfolding. In *Formal Methods in System Design 6 (1)*, pages 45–65.
- McMillan, K. L. (2000). *The SMV system*. Carnegie Mellon University. Manual for SMV version 2.5.4.

- Merz, R. and Litz, L. (2000). Objektorientierte mathematische Modellierung. *Informatik Spektrum*, pages 90–99.
- Merz, S. (2001). Model checking: A tutorial overview. In Cassez, F., Jard, C., Rozoy, B., and Ryan, M. D., editors, *Modeling and Verification of Parallel Processes*, volume 2067 of *Lecture Notes in Computer Science*, pages 3–38. Springer.
- Meyer, B. (1992). *Eiffel: The Language*. Object-Oriented Series. Prentice Hall, New York, NY.
- Meyer, B. (1997). *Object-Oriented Software Construction, Second Edition*. The Object-Oriented Series. Prentice-Hall, Englewood Cliffs (NJ), USA.
- Michalewicz, Z. and Fogel, D. (2000). *How to solve it: Modern Heuristics*. Springer.
- Millington, D. and Stapleton, J. (1995). Special report: Developing a RAD Standard. In *IEEE Software*, volume 12(5).
- Milner, R. (1980). *A Calculus of Communicating Systems*, volume 92 of *Lecture Notes in Computer Science*. Springer.
- Milner, R. (1989). *Communication and Concurrency*. Prentice-Hall International, Englewood Cliffs.
- Misra, J. and Chandy, K. M. (1981). Proofs of networks of processes. *IEEE Transactions on Software Engineering*, 7(4):417–426.
- Modelica Design Group, T. (2000). Modelica – a unified object-oriented language for physical system modeling v1.4. <http://www.modelica.org>.
- Moody, J. O. and Antsaklis, P. J. (1998). *Supervisory Control of Discrete Event Systems Using Petri Nets*. Kluwer Academic Publishers.
- Moor, T. (1998). Event driven control of switched integrator systems. In *Proc. ADPM'98 (Automatisation des Processus Mixtes: les Systèmes Dynamiques Hybrides)*, pages 271–277, Reims, France.
- Moor, T. (2000). *Approximationsbasierter Entwurf diskreter Steuerungen für gemischtwertige Regelstrecken*, volume 2 of *Forschungsberichte aus dem Max-Planck-Institut für Dynamik komplexer technischer Systeme*. Shaker, Aachen, Germany. Also PhD thesis, Fachbereich Elektrotechnik, Universität der Bundeswehr Hamburg.
- Moor, T., Davoren, J. M., and Raisch, J. (2001a). Modular supervisory control of a class of hybrid systems in a behavioural framework. In *Proceedings of the European Control Conference 2001*, pages 870–875, Porto, Portugal.
- Moor, T. and Raisch, J. (1999a). Discrete control of switched linear systems. In *Proceedings of the European Control Conference 1999*, Karlsruhe, Germany.
- Moor, T. and Raisch, J. (1999b). Supervisory control of hybrid systems within a behavioural framework. *Systems and Control Letters*, 38:157–166.
- Moor, T. and Raisch, J. (2000). Approximation of multiple switched flow systems for the purpose of control synthesis. In *Proc. of the 39th International Conference on Decision and Control, CDC'00*. IEEE Press.
- Moor, T. and Raisch, J. (2002). Abstraction based supervisory controller synthesis for high order monotone continuous systems. In *this volume*.
- Moor, T., Raisch, J., and Davoren, J. M. (2001b). Computational advantages of a two-level hybrid control architecture. In *Proc. of the 40th International Conference on Decision and Control, CDC'2001*, pages 358–362. IEEE Press.

- Moor, T., Raisch, J., and O'Young, S. D. (1998). Supervisory control of hybrid systems via l -complete capproximations. In *Proc. WODES'98 - International Workshop on Discrete Event Systems, Cagliari, Italy*, pages 426–431. IEEE.
- Moor, T., Raisch, J., and O'Young, S. D. (2002). Discrete supervisory control of hybrid systems based on l -complete approximations. *Journal of Discrete Event Dynamic Systems*, 12:83–107.
- Moormann, D. (2001). *Automatisierte Modellbildung der Flugsystemdynamik*. PhD dissertation, Aachen Technical University (RWTH Aachen), Aachen, Germany. in German.
- Moormann, D., Mosterman, P., and Looye, G.-J. (1999). Object-Oriented Computational Model Building of Aircraft Flight Dynamics and Systems. *Aerospace Science and Technology*, 3:115–126.
- Mosterman, P. and Biswas, G. (1999). A Java implementation of an environment for hybrid modeling and simulation of physical systems. In *International Conference on Bond Graph Modeling (ICBGM '99)*, pages 157–162. San Francisco.
- Mosterman, P., Otter, M., and Elmqvist, H. (1998). Modeling Petri Nets as Local Constraint Equations for Hybrid Systems Using Modelica. In *Proceedings of SCS Summer Simulation Conference*, pages 314–319, Reno, Nevada.
- Mosterman, P., Remelhe, M. P., Engell, S., and Otter, M. (2002). Simulation for analysis of aircraft elevator feedback and redundancy control. In Engell, S., Frehse, G., and Schnieder, E., editors, *Modelling, Analysis, and Design of Hybrid Systems*. Springer.
- Mosterman, P. J. (1999). An overview of hybrid simulation phenomena and their support by simulation packages. In *Hybrid Systems Computation and Control (HSCC'99)*, LNCS 1569. Springer.
- Mosterman, P. J. (2000a). HYBRSIM - a modeling and simulation environment for hybrid bond graphs. *Journal of Systems and Control*.
- Mosterman, P. J. (2000b). Implicit modeling and simulation of discontinuities in physical system models. In Engell, S., Kowalewski, S., and Zaytoon, J., editors, *The 4th International Conference on Automation of Mixed Processes: Hybrid Dynamic Systems*, pages 35–40.
- Mosterman, P. J. (2001). MASIM. Technical Report DLR-IB, DLR Oberpfaffenhofen, Oberpfaffenhofen, Germany.
- Mosterman, P. J. and Biswas, G. (1995). Modeling discontinuous behavior with hybrid bond graphs. In *1995 International Workshop on Qualitative Reasoning*, pages 139–147, Amsterdam. University of Amsterdam.
- Müller, C. (2002). *Analyse und Synthese diskreter Steuerungen hybrider Systeme mit Petri-Netz-Zustandsraummodellen*, volume 930 of *Fortschritt-Berichte VDI Reihe 8*. VDI-Verlag, Düsseldorf, Germany.
- Müller, C., Orth, P., and Rake, H. (2001). Analyse und Synthese diskreter Steuerungen hybrider Systeme mit einem Petri-Netz-Zustandsraummodell. In Schnieder, E., editor, *Engineering komplexer Automatisierungssysteme*, EKA 2001, pages 113–131, Braunschweig, Germany.
- Müller, C. and Rake, H. (1999). Modellbildung und Analyse hybrider Systeme mit Petri-Netzen und geschalteten Differentialgleichungen. In Schnieder, E., ed-

- itor, *Entwicklung und Betrieb komplexer Automatisierungssysteme*, EKA '99, pages 233–246, Braunschweig, Germany.
- Müller, C. and Rake, H. (2000). A Petri Net-State-Model for the Analysis and the Control Synthesis of Hybrid Technical Systems. In *Proceedings Hybrid Dynamic Systems*, ADPM 2000.
- Müller, K. (1996). *Entwurf robuster Regelungen*. B.G. Teubner Stuttgart.
- Münnemann, A. and Enste, U. (2001). Systemtechnische Integration gehobener Regelungsverfahren. *atp - Automatisierungstechnische Praxis*, 43(7):40–48.
- Nagel, K. and Schreckenberg, M. (1992). A cellular automaton model for free-way traffic. *Journal Phys.*, 2:2221.
- Naur, P. (1966). Proof of algorithms by general snapshots. *BIT (Nordisk tidskrift for informationsbehandling)*, 6(4):310–316.
- Nenninger, G. (2001). *Modellbildung und Analyse hybrider dynamischer Systeme als Grundlage für den Entwurf hybrider Steuerungen*, volume 902 of *Fortschritt-Berichte VDI Reihe 8*. VDI-Verlag.
- Nenninger, G., Frehse, G., and Krebs, V. (2000). Hybrid regions of attraction of piecewise affine hybrid systems. In *4th Conference on Automation of Mixed Processes: Hybrid Dynamic Systems ADPM 2000*, pages 87–92.
- Nenninger, G. and Krebs, V. (1998). Analysis of Hybrid Systems using Hybrid Dynamical Models. In *Hybrid Dynamical Systems: 3rd International Conference on Automation of Mixed Processes*, pages 428–431.
- Nenninger, G., Schnabel, M., and Krebs, V. (1999). Modellierung, Simulation und Analyse hybrider dynamischer Systeme mit Netz-Zustands-Modellen. *Automatisierungstechnik*, 47(3):118–126.
- Nenninger, G. M., Nixdorf, B., Krebs, V. G., and Lunze, J. (2001). Erreichbarkeitsanalyse hybrider Systeme. *at - Automatisierungstechnik*, 49(2):75–85.
- Nerode, A. and Kohn, W. (1993). Models for hybrid systems: Automata, topologies, controllability, observability. In Grossmann, R., Nerode, A., Ravn, A., and Rischel, H., editors, *Lecture Notes in Computer Science: Hybrid Systems*, volume 736, pages 317–356. Springer.
- Nicol, D. M. and Miner, A. S. (1995). The fluid stochastic petri net simulator. In *Proc. Sixth International Workshop on Petri Nets and Performance Models - PNPM'95*, pages 214–215, Durham, North Carolina, USA. IEEE-CS Press.
- Nicollin, X., Olivero, A., Sifakis, J., and Yovine, S. (1992). An approach to the description and analysis of hybrid systems. In *Proceedings of Workshop on Theory of Hybrid Systems*, volume 736 of *Lecture Notes in Computer Science*, pages 149–178, Lyngby, Denmark. Springer.
- Ning, B. (1998). Absolute braking and relative distance braking train operation control modes in moving block systems. In *Computers in Railways VI*, pages 991–1001. WIT Press, Lisbon.
- Nixdorf, B. and Lunze, J. (2000a). Control of a manufacturing cell. Technical report, Arbeitsbereich Regelungstechnik, TU Hamburg-Harburg. Internal document.
- Nixdorf, B. and Lunze, J. (2000b). KONDISK benchmark of an automated manufacturing cell. Technical report, Technical University of Hamburg-Harburg. (in German).

- Nordwig, A. (2000). the zooed homepage. Technische Universität Berlin. ISTI. <http://swt.cs.tu-berlin.de>.
- Nordwig, A. (2002). Formal integration of structural dynamics into the object-oriented modeling of hybrid systems. In *Proceedings of the European Simulation Multi-Conference '02*. to appear.
- Nöth, G. (1998). Randbedingungen für den Einsatz von regelungstechnischen Methoden. In *GMA-Kongress'98 Meß- und Automatisierungstechnik, VDI Bericht 1397*. VDI-Verlag.
- Nytsch-Geusen, C. (2001). *Berechnung und Verbesserung der Energieeffizienz von Gebäuden und ihren energietechnischen Anlagen in einer objektorientierten Simulationsumgebung*. PhD thesis, TU Berlin.
- Olivero, A. and Yovine, S. (1993). *KRONOS: A Tool for Verifying Real-Time Systems. User's Guide and Reference Manual*. Verimag, Grenoble, France.
- Omata, T. and Farooqi, M. A. (1996). Regrasps by a Multifingered Hand Based on Primitives. In *Proceedings of the IEEE International Conference on Robotics and Automation ICRA*, pages 2774–2780, Minneapolis, Minnesota, USA.
- Osder, S. (1999). Practical view of redundancy management application and theory. *Journal of Guidance, Control, and Dynamics*, 22(1):12–21.
- Otter, M., Elmqvist, H., and Mattson, S. (1999). Hybrid modeling in Modelica based on the synchronous data flow principle. In *CACSD'99*, Hawaii, USA.
- Otter, M., Remelhe, M., Engell, S., and Mosterman, P. (2000). Hybrid models of physical systems and discrete controllers. *at-Automatisierungstechnik*, 48:426–437.
- Owicki, S. S. and Gries, D. (1976). An axiomatic proof technique for parallel programs I. *Acta Informatica*, 6:319–340.
- Pachl, J. (1999). *Systemtechnik des Schienenverkehrs*. B. G. Teubner, Stuttgart.
- Panreck, K. (1999). Systembeschreibungen zur Modellierung komplexer Systeme. *at - Automatisierungstechnik*, 47(4):157.
- Park, T. and Barton, P. (1997). Implicit model checking of logic based control systems. *AIChE Journal*, 43(9):2246–2260.
- Pawletta, T. and Lampe, B. (2001). KONDISK project report no. Ia 724/8 – 2 — Modeling and simulation of modular-hierarchical systems with discrete event oriented structure dynamics. Technical report, University of Rostock. (in German).
- Pawletta, T., Lampe, B., Pawletta, S., and Drewelow, W. (1996). An object oriented framework for modeling and simulation of variable structure systems. In Ingalls, V., Cynamon, J., and Saylor, A., editors, *SCS Summer Simulation Conf., Portland, Oregon*, pages 8–13. SCS International.
- Pawletta, T., Lampe, B., Pawletta, S., and Drewelow, W. (2002). A DEVS-based approach for modeling and simulation of structure dynamics in hybrid systems. In Engell, S., Frehse, G., and Schnieder, E., editors, *Modelling, Analysis, and Design of Hybrid Systems*, Lecture Notes in Control and Information Science. Springer. (This volume).
- Pawletta, T., Lampe, B., Pawletta, S., Drewelow, W., and Schildmann, P. (2001). Modeling of temporal objects with self-dynamics in hybrid systems. In Panreck, K. and Dörrscheidt, F., editors, *15th Symp. of Simulation, Paderborn*,

- Frontiers in Simulation, pages 73–78, Ghent, Belgium. SCS Publishing House. (in German).
- Pawletta, T., Pawletta, S., and Dimitrov, E. (1994). Modeling and simulation of structure variable systems. In Kampe, G. and Zeitz, M., editors, *Progress in Simulation*, pages 59–64. Vieweg Publisher. (in German).
- Pawletta, T., Pawletta, S., Schildmann, P., and Drewelow, W. (1997). Interactive modeling and simulation of time-invariant system structures. In Kuhn, A. and Wenzel, S., editors, *Progress in Simulation*, pages 649–655. Vieweg Publisher. (in German).
- Paynter, H. M. (1961). Analysis and design of engineering systems. *The M.I.T. Press, Cambridge, Massachusetts*.
- Pearson, R. (1984). Modern control: Why don't we use it? *InTech*, 11:47–49.
- Pereira Remelhe, M., Deparade, A., and Engell, S. (2001). Integration und Synchronisierung von diskreten Beschreibungsformen und kontinuierlichen Systemmodellen in Modelica. In Panreck, K. and Dörrscheidt, F., editors, *Simulationstechnik, ASIM 2001, 15. Symposium*, pages 95–100. ASIM, SCS.
- Péter, I., Pretschner, A., and Stauner, T. (2000). Heterogeneous development of hybrid systems. In *Proc. GI workshop Rigorose Entwicklung software-intensiver Systeme*, pages 83–93.
- Petri, C. (1962). Kommunikation mit automaten. Technical Report 2, Institut für Instrumentelle Mathematik, Bonn. Schriften des IIM.
- Petterson, S. (1999). *Analysis and Design of Hybrid Systems*. PhD thesis, Chalmers University of Technology.
- Petzold, L. R. (1982). A description of DASSL: A differential/algebraic system solver. Technical Report SAND82-8637, Sandia National Laboratories, Livermore, California.
- Philips, P. (2001). *Modeling, Control and Fault Detection of Discretely Observed Systems*. PhD thesis, TU Eindhoven.
- Philips, P., Weiss, M., and Preisig, H. A. (1999). Control based on discrete-event models of continuous systems. In *Proceedings of the European Control Conference 1999*.
- Plank, J. (1997). *State Events in Continuous Modelling and Simulation*. PhD thesis, Technical University of Vienna.
- PNO (1999). Profibus-pa profile for process control devices, revision 3.0. Technical report, PNO, Karlsruhe.
- Pnueli, A. (1977). The temporal logic of programs. In *Proceedings of the 18th IEEE Symposium on Foundations of Computer Science (FOCS 1977)*, pages 46–57.
- Pnueli, A. (1981). The temporal logic of concurrent programs. *Theoretical Computer Science*, 13:45–60.
- Pnueli, A. (1984). In transition for global to modular temporal reasoning about programs. In *Logics and Models of Concurrent Systems*, volume 13 of *NATO ASI-F*. Springer.
- Pnueli, A. and Sifakis, J. (1995). Special issue on hybrid systems. *Theoretical Computer Science*, 138:1–239.
- Prähofer, H. (1991). *System Theoretic Foundations for Combined Discrete-Continuous System Simulation*. PhD thesis, Johannes Kepler University of Linz.

- Prähofer, H. (1996). An environment for DEVS-based multi-formalism modeling und simulation in C++. In *6th Annual Conference on AI, Simulation and Planning in High Autonomy Systems*, page 8pp. SCS International, San Diego.
- Prähofer, H. and Zeigler, B. (1992). Modelling and simulation. In Pichler, F. and Schwaertzel, H., editors, *CAST - Methods in Modelling*, pages 123–241. Springer Publisher, Berlin.
- Pretschner, A. (2001). Classical search strategies for test case generation with Constraint Logic Programming. In *Proc. Formal Approaches to Testing of Software*, pages 47–60.
- Pretschner, A., Lötzbeyer, H., and Philipps, J. (2001). Model Based Testing in Evolutionary Software Development. In *Proc. 11th IEEE Intl. Workshop on Rapid System Prototyping*, pages 155–160.
- Pretschner, A., Slotosch, O., and Stauner, T. (2000). Developing Correct Safety Critical, Hybrid, Embedded Systems. In *Proc. New Information Processing Techniques for Military Systems, NATO Research*.
- Preußig, J. (2000). *Formale Überprüfung der Korrektheit von Steuerungen mittels rektangulärer Automaten*. PhD thesis, Department of Chemical Engineering, University of Dortmund, Germany. (in German).
- Preußig, J., Kowalewski, S., Henzinger, T., and Wong-Toi, H. (1998). An algorithm for the approximate analysis of simple rectangular automata. In *Proc. 5th Int. School and Symposium on Formal Techniques in Fault Tolerant and Real Time Systems, Lyngby, Denmark, 1998*, Lecture Notes in Computer Science 1486, pages 228–240. Springer.
- Preußig, J., Stursberg, O., and Kowalewski, S. (1999). Reachability analysis of a class of switched continuous systems by integrating rectangular approximation and rectangular analysis. In Vaandrager, F. W. and van Schuppen, J. H., editors, *Hybrid Systems: Computation and Control, Proc. 2nd Int. Workshop, HSCC'99, Berg en Dal, The Netherlands, March 1999*, Lecture Notes in Computer Science 1569, pages 209–222. Springer.
- Preußig, J. and Wong-Toi, H. (2000). An procedure for the reachability analysis of rectangular automata. In *Proc. American Control Conference*, pages 1674–1678.
- Queille, J.-P. and Sifakis, J. (1982). Specification and verification of concurrent systems in CESAR. In Dezani-Ciancaglini, M. and Montanari, U., editors, *Proceedings of the 5th International Symposium on Programming, Turin, April 6–8, 1982*, pages 337–350. Springer.
- Raisch, J. (1998). A hierarchy of discrete abstractions for a hybrid plant. *JESA - European Journal of Automation, Special Issue on Hybrid Dynamical Systems*, 32(9-10):1073–1095.
- Raisch, J. (2000a). Complex systems – simple models? In *Proc. ADCHEM2000 - International Symposium on Advanced Control of Chemical Processes, Pisa*, pages 275–286.
- Raisch, J. (2000b). Discrete abstractions of continuous systems - an input/output point of view. *Mathematical and Computer Modelling of Dynamical Systems*, 6(1):6–29.

- Raisch, J., Itigin, A., and Moor, T. (2001). Hierarchical strategies for hybrid process control problems. In *Proceedings of the European Control Conference 2001*, pages 2534–2539, Porto, Portugal.
- Raisch, J. and Itigin, A. (2000). Synthesis of hierarchical process control systems based on sequential aggregation. In *Proc. 3rd Mathmod, Vienna*, pages 385–389.
- Raisch, J., Itigin, A., and Moor, T. (2000). Hierarchical control of hybrid systems. In Engell, S., Kowalewski, S., and Zaytoon, J., editors, *Proc. 4th International Conference on Automation of Mixed Processes: Dynamic Hybrid Systems*, pages 67–72. Shaker.
- Raisch, J., Klein, E., O’Young, S. D., Meder, C., and Itigin, A. (1998). Approximating automata and discrete control for continuous systems - two examples from process control. In Antsaklis, P., Kohn, W., Nerode, A., and Sastry, S., editors, *Hybrid Systems V*, LNCS 1567, pages 279–303. Springer.
- Raisch, J. and O’Young, S. (1997). A totally ordered set of discrete abstractions for a given hybrid or continuous system. In Antsaklis, P., Kohn, W., Nerode, A., and Sastry, S., editors, *Hybrid Systems IV*, volume 1273 of *LNCS*, pages 342–360. Springer.
- Raisch, J. and O’Young, S. D. (1998). Discrete approximation and supervisory control of continuous systems. *IEEE Transactions on Automatic Control, Special issue on hybrid systems*, 43:569–573.
- Rakoto-Ravalontsalama, N. and Aguilar-Martin, J. (1998). Diagnosing uncertain parameters to improve hybrid process model. In *Hybrid Dynamical Systems. 3rd International Conference on Automation of Mixed Processes*, pages 49–53, Reims.
- Ramadge, P. J. and Wonham, W. M. (1987). Supervisory control of a class of discrete event systems. *SIAM J. Control and Optimization*, 25:206–230.
- Ramadge, P. J. and Wonham, W. M. (1989). The control of discrete event systems. *Proceedings of the IEEE*, 77:81–98.
- Rational (1999). *Unified Modeling Language*. Rational Software Corporation. Version 1.3.
- Rational UML (1997). Unified modeling language, version 1.1. Rational Software Corporation.
- Rausch, M. and Hanisch, H.-M. (1995). Netz-Condition/Event-Systeme. In Schnieder, E., editor, *Entwurf komplexer Automatisierungssysteme - Methoden, Anwendungen und Tools auf der Basis von Petrinetzen und anderer formaler Beschreibungsmittel*, pages 55–71, Braunschweig.
- Raymond, P., Weber, D., Nicollin, X., and Halbwachs, N. (1998). Automatic testing of reactive systems. In *Proc. 19th IEEE Real-Time Systems Symposium*.
- Rebolledo, M. (2002). Development of a Concept for the Handling of Vagueness in the SQMA Modeling Approach. Diplomarbeit, Institut für Automatisierungs- und Softwaretechnik (IAS), Universität Stuttgart.
- Reckdahl, K. J. and Mitiguy, P. C. (1996). *AUTOLEV 3 Tutorial*. OnLine Dynamics, Inc., Sunnyvale, USA.
- Reisig, W. (1985). *Petri Nets, An Introduction*. EATCS, Monographs on Theoretical Computer Science. Springer, Berlin.

- Ricker, S. L., Sarkar, N., and Rudie, K. (1996). A Discrete-Event Systems Approach to Modeling Dextrous Manipulation. *Robotica*, 14:515–525.
- Royce, W. W. (1970). Managing the development of large software systems: Concepts and techniques. In *Proc. IEEE WESTCON*.
- Ruhl, H. (1999). Konzeption und Implementierung einer Visualisierungssoftware für den Modellprozess "Drei-Tank-System". Diplomarbeit, Institut für Automatisierungs- und Softwaretechnik (IAS), Universität Stuttgart.
- Rumbaugh, J. (1991). *Object-Oriented Modeling and Design*. Prentice-Hall Inc., New Jersey.
- Ruspini, D. and Khatib, O. (2000). A Framework for Multi-Contact Multi-Body Dynamic Simulation and Haptic Display. In *Proceedings of the 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Takamatsu, Japan.
- Schätz, B. and Pretschner, A. (2002). Model based development of embedded systems. Submitted to Model-Driven Approaches to Software Development, OOIS'02.
- Schildmann, P. (2000). Benchmarks for the simulator prototype MATSIM-2. Technical report, University of Rostock. (in German).
- Schiller, F. (1997). *Diagnose dynamischer Systeme auf der Grundlage einer qualitativen Prozessbeschreibung*. Dissertation, TU Hamburg-Harburg.
- Schlegl, T. (2002). *Diskret-kontinuierliche Regelung mehrfingriger Roboterhände zur robusten Manipulation von Objekten*. Number 928 in Fortschrittsberichte VDI, Reihe 8: Meß-, Steuerungs- und Regelungstechnik. VDI-Verlag, Düsseldorf.
- Schlegl, T., Buss, M., Omata, T., and Schmidt, G. (2001). Fast Dextrous Regrasping with Optimal Contact Forces and Contact Sensor Based Impedance Control. In *Proceedings of the IEEE International Conference on Robotics and Automation ICRA*, pages 103–107, Seoul, Korea.
- Schlegl, T., Buss, M., and Schmidt, G. (1997). Development of numerical integration methods for hybrid (discrete-continuous) dynamical systems. In *Proceedings of the IEEE/ASME International Conference on Advanced Intelligent Mechatronics AIM'97 (Tokyo, Japan, Paper No. 154)*.
- Schlegl, T., Buss, M., and Schmidt, G. (2002a). A Hybrid Systems Approach towards Modeling and Dynamical Simulation of Dextrous Manipulation. *IEEE Transactions on Mechatronics*, under review.
- Schlegl, T., Buss, M., and Schmidt, G. (2002b). Hybrid control of multi-fingered dextrous hands. *This volume*.
- Schlegl, T., Schnabel, M. K., Buss, M., and Krebs, V. G. (2000). State Reconstruction and Error Compensation in Discrete-Continuous Control Systems. *at - Automatisierungstechnik*, 48(9):439–447.
- Schnabel, M. (2001). *Diskret-kontinuierliche dynamische Systeme: Steuerung und Beobachtung*, volume 900 of *Fortschritt-Berichte VDI Reihe 8*. VDI-Verlag.
- Schöneburg, E., Heinzmann, F., and Feddersen, S. (1996). *Genetische Algorithmen und Evolutionsstrategien*. Addison-Wesley.
- Schröder, J. (2002). *Modelling, State Observation and Diagnosis of Quantised Systems*. Lecture Notes in Control and Information Sciences. Springer, Berlin.

- Schuler, H. (1992). Was behindert den praktischen Einsatz moderner regelungstechnischer Methoden in der Prozessindustrie? *atp - Automatisierungstechnische Praxis*, 34(3):116–123.
- Schumacher, J., Morse, A., Pantelides, C., and Sastry, S., editors (1999). *Special Issue on Hybrid Systems*, volume 35 of *Automatica*.
- Schürr, A. (1994). Logic based structure rewriting systems. In *Lecture Notes in Computer Science*. Springer.
- Schütt, H. (1990). *Entwicklung und Erprobung eines sehr schnellen, bitorientierten Verkehrssimulationssystems für Straßennetze*. PhD thesis, TU Hamburg-Harburg.
- SDL92 (1992). Specification and Description Language SDL, blue book. CCITT Recommendation Z.100.
- Seebeck, J. (1998). *Modellierung der Redundanzverwaltung von Flugzeugen am Beispiel des ATD durch Petrinetze und Umsetzung der Schaltlogik in C-Code zur Simulationssteuerung*. Diplomarbeit, Arbeitsbereich Flugzeugsystemtechnik, Technische Universität Hamburg-Harburg.
- Seiche, W. (1991). *Analyse und Synthese diskret gesteuerter Systeme mit Petri-Netzen*, volume 269 of *Fortschritt-Berichte VDI Reihe 8*. VDI-Verlag, Düsseldorf, Germany.
- Seiche, W. and Abel, D. (1993). Entwurf verklemmungsfreier Steuerungen auf der Grundlage einer graphentheoretischen Petri-Netz-Analyse. *Automatisierungstechnik*, 41:88–93.
- Selic, B., Gullekson, G., and Ward, P. T. (1994). *Real-Time Object-Oriented Modeling*. John Wiley & Sons Ltd, Chichester.
- Simon, C. (2001a). Developing software controllers with petri nets and a logic of actions. In *IEEE International Conference on Robotics and Automation, ICRA 2001*, Seoul, Korea.
- Simon, C. (2001b). *A Logic of Actions and Its Application to the Development of Programmable Controllers*. PhD thesis, Universität Koblenz-Landau.
- Simon, C., Ridder, H., and Marx, T. (1997). The petri net tools neptun and poseidon. *Fachberichte Informatik 15–97*, Universität Koblenz-Landau, Institut für Informatik, Rheinau 1, D-56075 Koblenz.
- Simon, C. and Thieme, J. (1999). Transformation zeitbewerteter Netzmodelle. *Fachberichte Fakultät Elektrotechnik 3–99*, Otto-von-Guericke-Universität Magdeburg, Institut für Automatisierungstechnik, Postfach 4120, D-39016 Magdeburg.
- Six, J. (1996). Abstandhaltung und Streckenleistungsfähigkeit. *Signal+Draht*.
- Smith, H. (1995). *Monotone Dynamical Systems*. American Mathematical Society, Providence.
- Sreenivas, R. S. and Krogh, B. H. (1991a). On condition/event systems with discrete state realizations. *Discrete Event Dynamic Systems: Theory and Application 1*, pages 209–236.
- Sreenivas, R. S. and Krogh, B. H. (1991b). Petri net based models for condition/event systems. *Proceedings of 1991 American Control Conference*, 3:2899–2904.

- Stahl, K. (1998). Comparing the expressiveness of different real-time models. Master's thesis, Christian-Albrechts-University of Kiel.
- Stauner, T. (2001). *Systematic development of hybrid systems*. PhD thesis, Technische Universität München.
- Stauner, T. (2002). Discrete-Time Refinement of Hybrid Automata. In *Proc. HSCC'02*. To be published.
- Stauner, T., Pretschner, A., and Péter, I. (2001). Approaching a Discrete-Continuous UML: Tool Support and Formalization. In *Proc. UML'2001 workshop on Practical UML-Based Rigorous Development Methods*, pages 242–257.
- Steffen, T. (2001). Rekonfiguration linearer Systeme durch eine Ergänzung des Reglers. Technical report, Ruhr University Bochum, Institute for Automation and Computer Control.
- Strikwerda, J. C. (1989). *Finite Difference Schemes and Partial Differential Equations*. Wadsworth & Brooks/Cole.
- Stursberg, O. (2000a). *Analyse gesteuerter verfahrenstechnischer Prozesse durch Diskretisierung*. PhD thesis, Department of Chemical Engineering, University of Dortmund, Germany. (in German).
- Stursberg, O. (2000b). Analysis of switched continuous systems based on discrete approximation. In *Proc. 4th Int. Conf. on Automation of Mixed Processes*, pages 73–78.
- Stursberg, O. and Engell, S. (2001). Optimized startup-procedures of processing systems. In *Proc. 6th IFAC Symp. Dynamics and Control of Process Sys.*, pages 231–236.
- Stursberg, O. and Engell, S. (to appear in July 2002). Optimal control of switched continuous systems using mixed-integer programming. In *Proc. 15th IFAC World Congress on Automatic Control*.
- Stursberg, O. and Kowalewski, S. (1999). Approximating switched continuous systems by rectangular automata. In *Proc. European Control Conference*. CD-ROM, file 1014–4.
- Stursberg, O. and Kowalewski, S. (2000). Analysis of controlled hybrid processing systems based on approximation by timed automata using interval arithmetics. In *Proc. 8th IEEE Mediterranean Conference on Control and Automation*. CD-ROM, file TA1–3.
- Stursberg, O., Kowalewski, S., and Engell, S. (2000). On the generation of timed discrete approximations for continuous systems. *Mathematical and Computer Modelling of Dynamical Systems*, 6(1):51–70. Special Issue on "Discrete Event Models of Continuous Systems".
- Stursberg, O., Kowalewski, S., Hoffmann, I., and Preußig, J. (1997). Comparing timed and hybrid automata as approximations of continuous systems. In Antsaklis, P., Kohn, W., Nerode, A., and Sastry, S., editors, *Hybrid Systems IV*, volume 1273 of *LNCS*, pages 361–377. Springer.
- Stursberg, O. and Panek, S. (to appear in 2002). Control of switched continuous systems based on disjunctive formulations. In *5th Int. Workshop on Hybrid Systems: Computation and Control*, LNCS. Springer.
- Sussmann, H. (1999). A maximum principle for hybrid optimal control problems. In *Proc. 38th IEEE Conf. Decision and Control*, pages 425–430.

- Tavernini, L. (1987). Differential automata and their discrete simulators. *Nonlinear Analysis, Theory, Methods, and Applications*, 11:665–683.
- Thieme, J. (2002). *Symbolische Erreichbarkeitsanalyse und automatische Implementierung struktureller, zeitbewerteter Steuerungsmodelle*. PhD thesis, Martin-Luther-Universität Halle-Wittenberg, Mathematisch-Naturwissenschaftlich-Technische Fakultät.
- Thieme, J. and Lüder, A. (1999). Transformation von Netzmodellen zur Analyse technischer Systeme. Fachberichte Fakultät Elektrotechnik 2–99, Otto-von-Guericke-Universität Magdeburg, Institut für Automatisierungstechnik, Postfach 4120, D-39016 Magdeburg.
- Thomas, C. (1996). *An Object Oriented Approach to Modeling and Simulation of Complex Systems*. VDI-Verlag. (in German).
- Thomas, J. (1995). *Numerical Partial Differential Equations: Finite Difference Methods*. Springer.
- Tittus, M., Egardt, B., and Lennartson, B. (1994). Hybrid systems in process control. In *3rd IEEE Conference on Decision and Control*, pages 3587–3595.
- Tomlin, C. (1999). Towards efficient computation of solutions to hybrid systems. In *Proceedings of the 38th IEEE Conference on Decision and Control (Phoenix, AZ)*, pages 3532–3537.
- Tomlin, C. and Greenstreet, M. R., editors (2002). *Hybrid Systems: Computation and Control, 5th International Workshop (HSCC'02)*, volume 2289 of *Lecture Notes in Computer Science*, Stanford, CA, USA. Springer.
- Tomlin, C., Lygeros, J., and Sastry, S. (2000). A game theoretic approach to controller design for hybrid systems. *Proceedings of the IEEE*, 88(7):949–970.
- Treseler, H. (2001). *Ein Rechnerwerkzeug zur formalen Verifikation diskret gesteuerter verfahrenstechnischer Prozesse*. PhD thesis, Department of Chemical Engineering, University of Dortmund, Germany. (in German).
- Trontis, A. and Spathopoulos, M. (2001). Target control for hybrid systems with linear continuous dynamics. In *Proc. 40th IEEE Conf. on Decision and Control*, pages 1229–1234.
- Turing, A. M. (1949). On checking a large routine. In *Report of a Conference on High Speed Automatic Calculating Machines*, pages 67–69, Cambridge. University Mathematics Laboratory.
- Uebel, H. (2000). Durchsatz von Strecken und Stationen bei Bahnen. In *Gesamtverkehrsforum 2000*, number 1545 in VDI Berichte, pages 257–275. VDI-Verlag, Düsseldorf.
- Uhrmacher, A. M. and Arnold, R. (1994). Distributing and maintaining knowledge: Agents in variable structure environment. In *5th Annual Conference on AI, Simulation and Planning in High Autonomy Systems*, pages 178–194.
- Utkin, V. (1992). *Sliding Modes in Control Optimization*. Springer.
- Vaandrager, F. and van Schuppen, J., editors (1999). *Hybrid Systems – Computation and Control, Proc. 2nd Int. Workshop HSCC'99, Berg en Dal, The Netherlands, March 1999*, volume 1569 of *Lecture Notes in Computer Science*. Springer.
- Valavanis, K. (1997). Special issue on applications of discrete event and hybrid systems. *IEEE Robotics and Automation Magazine*, 4.

- van der Schaft, A. and Schumacher, H. (2000). *An Introduction to Hybrid Systems*, volume 251 of *Lecture Notes in Control and Information Science*. Springer, London.
- van der Schaft, A. J. and Schumacher, J. M. (1996). The complementary-slackness of hybrid systems. *Math. Contr. Signals Syst.*, 9:266–301.
- Vardi, M. Y. and Wolper, P. (1994). Reasoning about infinite computations. *Information and Computation*, 115(1):1–37.
- Vecchiotti, A. and Grossmann, I. (1999). Logmip: A disjunctive 0-1 nonlinear optimizer for process system models. *Comp. and Chemical. Eng.*, 23:555–565.
- Verghese, G. C., Lévy, B. C., and Kailath, T. (1981). A generalized state-space for singular systems. *IEEE Transactions on Automatic Control*, 26(4):811–831.
- Vidal, R. (1993). *Applied Simulated Annealing*. Springer, Berlin.
- Vidal, R., Schaffert, S., Shakernia, O., Pappas, G., and Sastry, S. (2001). Decidable and semi-decidable controller synthesis for classes of discrete-time hybrid systems. In *Proc. 40th IEEE Conf. Decision and Control*, pages 1243–1248.
- Visual Object Net ++ (2000). http://www.r-drath.de/VON/von_e.htm.
- von Stryk, O. (2000). Numerical hybrid optimal control and related topics. Habilitation Dissertation, Technische Universität München.
- von Stryk, O. (2001). User's guide for DIRCOL version 2.1: A direct collocation method for the numerical solution of optimal control problems. Technical report, Simulation and Systems Optimization Group, Technische Universität Darmstadt. WWW: www.sim.informatik.tu-darmstadt.de/sw/.
- von Stryk, O. and Bulirsch, R. (1992). Direct and indirect methods for trajectory optimization. *Annals of Operations Research*, 36:357–373.
- von Stryk, O. and Glocker, M. (2000). Decomposition of mixed-integer optimal control problems using branch and bound and sparse direct collocation. In *ADPM – 4th Int'l Conf. on Automation of Mixed Processes: Hybrid Dynamic Systems*, pages 99–104.
- von Stryk, O. and Glocker, M. (2001). Numerical mixed-integer optimal control and motorized traveling salesmen problems. *APII – JESA (Journal européen des systèmes automatisés – European Journal of Control)*, 35(4):519–533.
- Vries, R. d., Tretmans, J., Belinfante, A., Feenstra, J., Feijs, L., Mauw, S., Goga, N., Heerink, L., and Heer, A. d. (2000). Côte de Resyste in Progress. In *Progress 2000 – Workshop on Embedded Systems*, pages 141–148.
- W3C (1998). Extensible markup language XML. <http://www.w3.org/TR/REC-xml>.
- Wiedemann, R. (1974). Simulation des Straßenverkehrsflusses. Technical Report 8, Instituts für Verkehrswesen der Universität, Karlsruhe, Germany.
- Wiedemann, R. (1991). Modelling of rti-elements on multi-lane roads. In of the European Community, C., editor, *Advanced Telematics in Road Transport*, Brussels.
- Wieting, R. (1996). Modeling and simulation of hybrid systems using hybrid high-level nets. In *8th European Simulation Symposium ESS'96*, volume 1, pages 96–100.
- Wieting, R. (1998). *Modellbildung und Simulation mit hybriden höheren Netzen*. PhD thesis, Carl von Ossietzky Universität, Oldenburg. ISBN 3-8265-3291-0.

- Willems, J. C. (1989). Models for dynamics. *Dynamics Reported*, 2:172–269.
- Willems, J. C. (1991). Paradigms and puzzles in the theory of dynamic systems. *IEEE Transactions on Automatic Control*, 36:258–294.
- Williams, H. P. (1978). *Model Building in Mathematical Programming*. J. Wiley P., 1st edition.
- Woelfl, K. (1995). *Planung von Manipulationsvorgängen einer Roboterhand*. Number 455 in Fortschrittsberichte VDI, Reihe 8: Meß-, Steuerungs- und Regelungstechnik. VDI-Verlag, Düsseldorf.
- Wolf, A. (2001). *Components and Interfaces for Modeling and Simulation of Continuous-Discrete Systems*. PhD thesis, Technical University of Magdeburg. (in German).
- Wöllhaf, K. (1995). *Object Oriented Modeling and Simulation of Multi-Product Batch Plants*. PhD thesis, University of Dortmund. (in German).
- Wolter, K. (1999). *Performance and Dependability Modelling with Second Order Fluid Stochastic Petri Nets*. Shaker, Aachen.
- Wolter, K. (2001). A performability model for a hybrid reactor system. In Djemame, K. and Kara, M., editors, *Proc. 17th annual UK Performance Engineering Workshop*, pages 13–22, Leeds, UK.
- Wolter, K. and Zisowsky, A. (2001). Performance evaluation. *On Markov Reward Modelling with FSPNs*, 44:165–186.
- Xu, X. and Antsaklis, P. (2001). An approach for solving general switched linear quadratic optimal control problems. In *Proc. 40th IEEE Conf. Decision and Control*, pages 2478–2483.
- Yovine, S. (1997). Kronos: a verification tool for real-time systems. *Software Tools for Technology Transfer*, 1(1,2):123–133.
- Zaytoon, J., editor (1998). *3rd Int. Conf. on Automation of Mixed Processes: Hybrid Dynamic Systems (ADPM'98)*, Reims, France. Université de Reims.
- Zeigler, B. (1976). *Theory of Modelling and Simulation*. John Wiley & Sons.
- Zeigler, B. (1984). *Multifaceted Modelling and Discrete Event Simulation*. Academic Press, Inc.
- Zeigler, B. (1990). *Object-Oriented Simulation with Hierarchical, Modular Models*. Academic Press, Inc.
- Zeigler, B. and Prähofer, H. (2000). *Theory of Modelling and Simulation*. Academic Press, London, second edition.
- Zhang, P. and Cassandras, C. (2001). An improved forward algorithm for optimal control of a class of hybrid systems. In *Proc. 40th IEEE Conf. Decision and Control*, pages 1235–1236.
- Zhivoglyadov, P. and Middleton, R. (1999). A novel approach to systematic switching control design for a class of hybrid systems. In *Proc. of the 38th International Conference on Decision and Control, CDC'99*. IEEE Press.
- Zhu, P. (2001). *Betriebliche Leistung von Bahnsystemen unter Störungsbedingungen*. VDI-Verlag, Düsseldorf.
- Zimmermann, A., German, R., Freiheit, J., and Hommel, G. (2000). Petri net modelling and performability evaluation with timenet 3.0. In *Proc. 11th Int. Conf. on Computer Performance Evaluation; Modelling Techniques and Tools*, number 1786 in LNCS, pages 188–202, Schaumburg, IL, USA.

Zisowsky, A. (1998). Entwurf und Implementierung eines Verfahrens für die transiente Analyse fluider stochastischer Petri-Netze. Master's thesis, TU Berlin.

Index

- ω -automata, 159
- θ -scheme, 197
- abstraction, 164, 236, 249, 271
- activator, 306
- additional firing condition, 309
- ADI method, 198
- advanced control, 61
- aircraft attitude control, 369
- aircraft elevator control, 373
- alternating direction implicit scheme, 198
- analysis, 156
- annealing furnace, 29
- approximate analysis, 166
- approximation, 252
- arbiter example, 11
- assignment, 218
- assume/guarantee, 241
- assumption/commitment, 238
- attributed hybrid dynamic nets, 27
- AutoFocus, 46
- automata
 - ω , 159
 - cellular, 422
 - discrete, 229
 - hybrid, 158, 230
 - nondeterministic, 75
 - rectangular, 165
 - stochastic, 77
 - stopwatch, 163
 - timed, 158, 230
- batch, 56
 - evaporator, 99
 - plant, 212
- Bellman, 273
- bisimulation, 179
- bond graph model, 384
- Branch-and-Bound, 322, 348
- branching time temporal logic, 233
- car diesel engine, 288
- cellular automata, 422
- Charon, 39
- charts
 - hybrid sequence, 42
 - message sequence, 45
 - object-oriented state, 146
- chemical reactor, 349
- component model, 53
- compositionality, 237
- computation
 - issues, 158
 - model, 231
- computational tree logic, 233
- computing model, 124
- condensation
 - of a graph, 301
 - of an evolution graph, 301
- constraint system, 218
- control
 - correction of, 305
 - design, 272, 342
 - hybrid, 176
 - linear, 275
 - optimal, 318
 - reconfiguration, 267
 - supervisory, 84, 249
 - synthesis, 286
 - via left eigenvector, 184
- controllability, 305
- controller synthesis
 - using verification, 286
- conveyor belt, 26
- CPLEX, 350
- Crank-Nicolson scheme, 197
- CTL, 233
- cycle, 300
- DAE
 - higher index, 383
- data structure, 236
- deactivator, 306
- deadlock, 160, 299
- decidability, 161, 179, 234
- decomposition, 11
- DES/M, 90
- distillation column, 260

- deterministic behaviour, 298
- diagnosis, 395
- diesel engine, 288
- Dirac impulse, 10
- Dircol, 320
- direct collocation, 320
- discrete
 - abstraction, 251
 - approximations, 74
 - automata, 229
 - boundary condition, 198, 200
 - control, 21, 295
 - control loop, 270
 - controller, 337
 - controller design, 272
 - error compensation, 455
 - model, 75
 - time, 341
- discretisation, 164, 193, 320
- disjunctive form, 345
- Dymola, 91
- dynamics
 - structural, 146
- eigenvector, 184
- error compensation, 455
- evolution graph, 299
- filtration process, 63
- firing
 - condition, additional, 305
 - sequence, 219
- flow, 158
- formal
 - methods, 225
 - verification, 227
- function blocks, 53
- genericity, 141
- guard, 158
- HDS, 313
- HSM, 442
- hybrid
 - automata, 158, 230, 339
 - control, 176, 317, 447
 - dynamic nets, 16
 - dynamical system, 313
 - object nets, 24
 - optimal control, 318
 - Petri Net, 356
 - phenomena, 5
 - reachability, 177
 - reachability graph, 295
 - sequence charts, 42
 - state, 160
 - state model, 314, 442
 - state vector, 297
 - token, 28
- hybrid system, 4
 - example, 26, 29, 43, 63, 99, 116, 167, 187, 201, 212, 260, 280, 288, 291, 297, 302, 324, 327, 349, 369, 409, 437
 - nature, 154
- HyCharts, 46
- HyROOM, 42
- HySC, 42
- Hytech, 162, 234
- IB-state, 298
- IMMA, 38
- impedance control, 449
- implicitness parameter, 197
- interval, 214, 217
- invariant, 158
- invariant behaviour, 298
- Java, 148
- Kripke structure, 231
- KRONOS, 162, 234
- laboratory batch plant, 212
- Langrange-multiplier, 446
- LD-systems, 181
- linear divided system, 181
- linear programming
 - mixed integer, 348
- linear time temporal logic, 233
- Lipschitz
 - condition, 5
 - constant, 258
- liveness, 301
- LTL, 233
- M-approach, 344
- manifolds
 - attractive, 259
- manufacturing cell, 116, 201, 291, 302

- MaSiEd, 41
- MATHEMATICA, 220
- Matlab, 128
 - Real-Time Workshop, 43
- MatrixX, 38
- maximal step, 298
- minimal extension to a control, 305
- model
 - based development, 39
 - checking, 228
 - discrete, 75
 - discrete-event, 270
 - transformation, 423
- Modelica, 90, 142
- modeling, 85
 - component-oriented, 395
 - environment, 86
 - frameworks, 154
 - hybrid systems, 154
 - qualitative, 397
- modular
 - hierarchical systems, 109
 - modelling, 296
 - verification, 237
- monotone systems, 253
- moving horizon, 348
- MSC, 45
- multi-arm transportation task, 324
- multi-fingered robotic hand, 439
- MVC, 147
- net elements, 18
- net state model, 174
- nondeterministic automata, 75
- NSM, 174
- object-oriented
 - modeling, 90, 377
 - structuring, 140
- online
 - analysis, 408
 - state space reduction, 403
- online monitoring, 394
- OOSC, 146
- optimisation, 273
- path quantifier, 233
- Peaceman-Rachford scheme, 200
- performance model, 193
- Petri Net, 295
 - coloured, 357
 - fluid stochastic, 193
 - hybrid, 356
 - Place/Transition, 296
 - State-Model, 296
 - stochastic, 193
 - timed coloured, 359
- place
 - complementary, 309
- POSEIDON, 216, 220
- process control, 56
- production unit, 291
- qualitative
 - monitoring and diagnosis, 395
- quantisation, 12, 71
 - boxes, 256
- quantised process model, 271
- quantiser, 271
- random flow, 195
- reachability, 161, 219
 - affine, 180
 - analysis, 157, 176
 - hybrid, 177
 - set, 219
- reachability graph, 195
- reconfiguration, 267, 268
 - linear, 275
- rectangular automata, 165
- redundancy, 380
- reflecting boundary, 195
- relaxation, 344
- requirements definition, 135
- reversibility of a hybrid system, 301
- robot, 327, 437
- ROOM, 41
- ROOMcharts, 41
- run, 159
- sampling, 69
- self-loop, 309
- Semi-Markov-process, 78
- sequential control, 295
- significant
 - firing condition, 306
 - place, 307
 - state, 307

- simulation, 85, 156, 361
 - modular, 123
 - monolithic, 123
- SMV, 234
- SPIN, 234
- SQMD, 395
- state
 - machine, 45
- state space
 - extended model, 297
 - model, 295
 - reduction, 403
- statechart, 96, 381
- stochastic
 - automata, 77
 - process, 195
- stopwatch automata, 163
- strong component, 301
- structural changes, 109
- structure
 - Kripke, 231
- supervisory control, 84, 249
- switched differential equations, 297
- symbolic
 - firing sequence, 219
 - marking, 217, 218
- synthesis of control corrections, 304
- system
 - first order, 20
 - second order, 20
- systems
 - monotone, 253
- template, 57
- temporal induction, 239
- temporal logic, 232
- temporal operator, 233
- term, 217
- three-tank-system, 409
- time interval, 214
- timed
 - automata, 158, 230
 - CP-net, 359
- timestamp
 - net, 214
- timewise stuck, 216
- titration plant, 280
- token, 28
- traffic modeling, 420
- transition
 - congruent, 309
 - critical, 306
- traveling salesman, 331
- two-tank-system, 167, 187, 297
- UML, 39
- underactuated robot arm, 327
- UPPAAL, 162, 234
- upwind scheme, 197
- utilization, 207
- V-model, 134
- variable structure systems, 109
- verification, 156, 227
 - compositional, 237
 - in controller synthesis, 286
- virtual actuator, 278
- wire stretching plant, 43